Treball de Fi de Grau

Grau d'Enginyeria en Tecnologies Industrials

Predicció del consum energètic emprant xarxes neuronals

MEMÒRIA

Autor: Ricardo Sebastián Sebastián
Director: Ramon Costa Castelló

Convocatòria: Gener 2020

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

L'objectiu del treball és l'obtenció d'una predicció fiable del consum elèctric d'un habitatge mitjançant les xarxes neuronals. Abans, però, és necessari realitzar una introducció als diferents conceptes teòrics associats per tal de facilitar la comprensió de la feina realitzada. Primer de tot, s'expliquen quins són els diferents elements que la configuren i quins són els seus principals paràmetres, dels quals s'ha estudiat la seva influència en la qualitat de les prediccions. També es presenten els principals problemes que poden aparèixer durant la creació de la xarxa, a més de quins són els diferents conjunts de dades que és necessari definir per a dur a terme l'estudi. Posteriorment, s'explica quina és la història tant de la intel·ligència artificial com de les xarxes neuronals, essencial per a comprendre quin és l'estat actual d'aquest camp.

Una vegada es realitza la introducció teòrica, s'explica quines dades de partida s'han emprat a les proves, a més de com s'han tractat per a posar-les a un format adient per ser introduïdes a la xarxa neuronal. També es detalla quina és l'estructura d'una xarxa neuronal creada utilitzant, a més del programari Python, les biblioteques Keras i Tensorflow.

A continuació es mostren els resultats de les dues proves principals i de les subproves associades a cadascuna d'elles. En elles es mostra la importància que té, entre d'altres factors, la importància del nombre de dades de les què es disposa i de la influència d'altres factors, com els referents a la data de la mostra o a les dades meteorològiques associades al dia al qual es va registrar el consum. Després, es valora l'impacte ambiental associat a aquest estudi, tant des del punt de vista de l'impacte generat durant la seva realització com de les conseqüències que pot portar l'aplicació d'aquest model a un cas real. En darrer lloc, s'avalua quin és el pressupost necessari per a dur a terme el projecte i les implicacions a nivell econòmic que comportaria la implantació del model.

Sumari

1. Introducció

Una aspecte fonamental de la societat actual és la utilització d'energia elèctrica. Vivim envoltats d'aparells que fan servir aquesta font d'alimentació i molts d'ells han esdevingut imprescindibles. Per a poder subministrar la quantitat d'electricitat que s'ajusti a les necessitats dels usuaris en cada moment, és essencial conèixer amb certa antelació quin serà el consum que es produirà en les properes hores. Això resulta útil per a poder estimar situacions a les quals la demanda és elevada i pot portar a fallades de la xarxa provocades per situacions extremes, com onades de calor o de fred. També és molt important en sistemes de tipus micro-xarxa, que poden funcionar de forma independent de la xarxa principal, on la seva escassa inèrcia (una caiguda en la producció d'energia pot provocar una caiguda gairebé instantània en el subministrament) fa necessari, a més d'estimar la producció d'energia, una predicció del perfil de consum, i aquestes també són útils per calcular les característiques dels dispositius d'emmagatzematge habituals en aquests tipus de xarxes.

1.1. Objectius del projecte

L'objectiu principal d'aquest projecte és l'obtenció de la predicció del consum elèctric d'un habitatge. Fent servir el programari Python, es processaran les dades de partida i es transformaran en un format adient per poder treballar amb elles. Amb el mateix software, es crearà una xarxa neuronal que permeti obtenir una estimació de la demanda energètica del pis. També s'estudiarà la influència en la precisió de les prediccions de la introducció de diferents paràmetres o dades addicionals, que poden ser de tipus temporal (per exemple, dia de la setmana o mes al qual es van recollir les dades de consum) o de caire meteorològic (és a dir, les temperatures registrades).

1.2. Abast del projecte

En aquest projecte l'estimació de la producció s'ha centrat en el costat de la demanda que fan els usuaris, tot estudiant el cas concret d'un habitatge. El principal motiu de la tria d'aquest cas ha estat el gran període de temps durant el qual es va portar a terme la recollida de consum elèctric, quatre anys i mig. Al tenir aquest nombre elevat de dades, les prediccions seran de major qualitat i les conclusions dels estudis seran més concloents. A

més, es disposa d'una informació gens habitual però alhora molt valuosa per tal d'interpretar tant les dades de partida com els resultats de cadascuna de les proves portades a terme amb la xarxa neuronal: els hàbits de consum dels residents al pis. Es faciliten els electrodomèstics i aparells elèctrics que són d'ús habitual o que poden tenir un impacte significatiu en la demanda d'energia, a més de com s'utilitzen al llarg del dia. Aquesta informació no sol ser visible per a altres tipus d'edificis que fan públiques les seves dades de consum.

2. Xarxes neuronals

Una xarxa neuronal és un algorisme inspirat en l'estructura del cervell humà que pretén imitar el seu funcionament a l'hora de reconèixer patrons. Aquesta tècnica té un gran nombre d'aplicacions, com poden ser el reconeixement d'imatges i de so, problemes de classificació o problemes de regressió (als quals s'ha de predir un valor o valors concrets).

2.1. Components bàsics i arquitectura de la xarxa

L'element bàsic d'una xarxa neuronal és la neurona (també coneguda com a node), la qual té diverses entrades i una única sortida. Les entrades es multipliquen per un valor numèric, conegut com pes. Després, tots aquests valors es sumen i el resultat s'introdueix a la funció d'activació, que proporciona una sortida depenent del tipus de funció emprada. Aquestes neurones s'organitzen en capes que es van succeint des de l'entrada de les dades fins a la sortida. En el tipus de xarxa utilitzada en aquest treball, coneguda com perceptró multicapa o *feedforward neural network,* un node d'una capa està connectat amb tots els de la capa següent. L'arquitectura bàsica de la xarxa és la següent:

- La capa d'entrada, també coneguda pel seu nom en anglès *input layer*, que és la que rep els valors a partir dels quals s'ha de fer la predicció.
- Les capes ocultes (*hidden layers*), ubicades entre la capa d'entrada i la de sortida.
- La capa de sortida (*output layer*), és la darrera capa i la que mostra els resultats.

El nombre de neurones de la primera capa ha de ser coincident amb el nombre de valors d'entrada. A la capa de sortida, el nombre de nodes ha de ser el del nombre de resultats que es volen obtenir. Les entrades i sortides estan contingudes en les *input matrix* i *target matrix,* respectivament. La primera s'utilitza per introduir-la a la xarxa neuronal i generar les prediccions a partir d'ella: la segona s'empra per comparar els valors reals amb als resultats obtinguts.

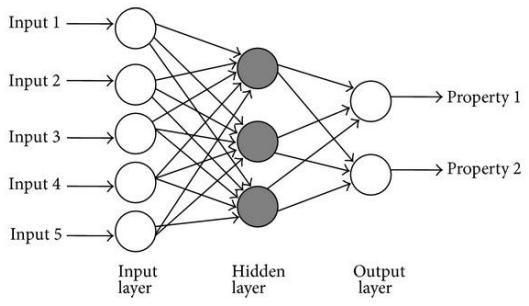

Figura 2.1. Estructura del perceptró multicapa. Font: [1]

Els pesos que tenen les neurones es van ajustant durant el procés d'entrenament, que consisteix en el següent: les dades passen per la xarxa neuronal i, una vegada obtinguts els resultats, es comparen amb els valors de la *target matrix* i, en funció d'això i depenent d'un paràmetre que s'explicarà més endavant, l'optimitzador, s'ajusten els pesos i es repeteix el procés fins que s'arriba al criteri de precisió fixat. El que succeeix dins de la xarxa neuronal no es sol considerar ni estudiar: es tracta com una caixa negra on durant les iteracions només interessen com són les característiques de la sortida.

2.2. Paràmetres de la xarxa

Al disseny de la xarxa, els paràmetres principals que s'han de seleccionar són els següents:

- Nombre de neurones a cada capa. Tot i que el nombre de neurones de la capa d'entrada i de sortida venen fixats pel nombre de valors que s'introdueixen i que es volen obtenir de la xarxa, el nombre de neurones a les capes ocultes és determinant, ja que d'aquest, entre d'altres paràmetres, depèn que es produeixin problemes com l'*underfitting* i *l'overfitting,* que s'explicaran més endavant.

- Nombre de capes ocultes. La majoria de xarxes neuronals utilitzades a l'actualitat solen tenir més d'una capa. Normalment, el nombre òptim de capes ocultes sol trobar-se entre 1 i 2, i no és freqüent que es necessitin més de 3 capes, tot i que a vegades poden ser necessàries per problemes complexos [2].

- La funció d'activació. S'inclou a cadascuna de les neurones que conformen la xarxa neuronal (excepte a la capa d'entrada). Aquestes funcions determinen quina és la sortida dels nodes a partir de la suma ponderada de les entrades pels pesos. Dues de les funcions actualment més utilitzades, i que són les que s'empraran al treball, són la funció ReLU a les capes ocultes i la funció sigmoide a la capa de sortida. La primera roman inactiva per a valors de la suma ponderada inferiors o iguals a zero (és a dir, el seu valor de sortida és zero) i es comporta de forma lineal a partir d'aquest valor. La funció sigmoide té els valors de sortida limitats al rang comprès entre 0 i 1, per tant, per si la suma ponderada és inferior a zero la sortida segueix sent zero, i per valors superiors a 1 la sortida és la unitat. A continuació es mostren els gràfics del comportament d'aquestes funcions:

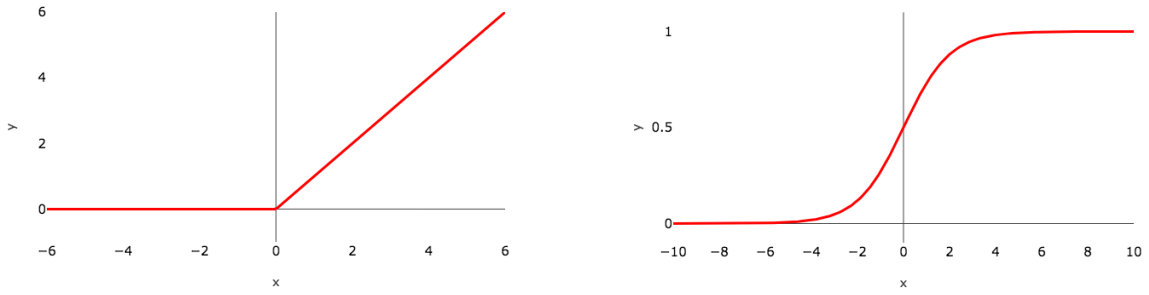

Figura 2.2. Funcions d'activació ReLU i Sigmoide. Font: [3][4]

- La *loss function,* o funció de pèrdua, és la funció que es vol minimitzar. S'ha de triar en funció del tipus de problema del què es tracti.

- La mètrica, que serveix per avaluar la precisió del model. Al igual que l'anterior, també s'ha de triar en funció del problema.

- L'optimitzador, també conegut com a algorisme d'optimització, és el què s'encarrega d'actualitzar els pesos de les neurones al final de cada iteració, i defineix com és l'estratègia que es segueix per fer-ho. En general, i als casos que es tractaran en aquesta memòria, el que fan els optimitzadors és calcular el gradient de la *loss function* i ajustar els pesos en la direcció de baixada del gradient. Hi ha infinitud d'algorismes que realitzen aquesta tasca: en aquest treball es compararan dos d'ells: un és SGD (*Stochastic Gradient Descent*), sigles en anglès de gradient de descens estocàstic, i Adam (*Adaptative Moment Estimation*). El seu funcionament i estructura es detallaran més endavant, quan es realitzi la comparació per seleccionar quin és el que proporciona millors resultats.

- La taxa d'aprenentatge. És un paràmetre usualment relacionat amb l'optimitzador, ja que aquest algorisme sol incloure associat un valor concret o bé una taxa variable. Controla amb com d'allunyat és el punt del gradient respecte al punt anterior, és a dir, la velocitat amb la qual s'ajusten els pesos de la xarxa neuronal. És un hiperparàmetre molt important, ja que és clau per a la convergència de l'algorisme. Una taxa molt petita pot fer que l'aprenentatge sigui molt lent, mentre que un valor molt elevat fa que es sobrepassi el valor mínim i aquest sigui pràcticament impossible de trobar [5].

2.3. Conjunts de dades

Per a entrenar una xarxa, és necessària la creació de tres conjunts amb les dades inicials de les què es disposa:

- El conjunt d'entrenament, que són les dades amb les quals s'ajusten els pesos de les neurones.

- El conjunt de validació, és el que s'utilitza per comprovar, al final de cada iteració, si amb els nous pesos calculats els resultats milloren, empitjoren o es mantenen similars. Aquestes dades sempre han de ser diferents a les que s'empren per l'entrenament. L'ús d'aquest conjunt és essencial per tal d'evitar l'*overfitting* i *l'underfitting,* dos problemes freqüents al camp del *machine learning* i que s'explicaran més endavant. A més, també és important per tal de calcular quan s'ha d'aturar el procés iteratiu, amb el que es coneix com a *early stopping:* es monitoritzen els valors de la *loss function* al conjunt de validació i es determina i s'indica el nombre d'etapes sense observar una millora en l'indicador a les quals s'ha d'aturar l'entrenament. A més, amb aquesta tècnica es restauren els pesos de la millor iteració. A cada una de les etapes d'aquest treball es defineixen quines són les condicions que marcaran la finalització de l'entrenament.

- El conjunt de test, usat per avaluar, al final de l'entrenament, com són els resultats, la *loss function* i la mètrica triada. És important que aquestes dades tinguin una distribució semblant a les del conjunt d'entrenament per tal què aquest sigui efectiu, per tant, és usual que als casos com el que és objecte d'estudi en aquest treball, on les dades siguin una sèrie temporal s'aleatoritzin les mostres per tal d'evitar aquestes diferències. Com només s'utilitza una vegada al acabar el procés iteratiu, les seves dades no s'han d'haver fet servir en cap altre dels conjunts.

2.4. Problemes associats: *Overfitting* i *underfitting*

Durant l'entrenament de la xarxa neuronal, i per diferents motius com els exposats als apartats anteriors, es pot produir *overfitting* i *underfitting.* És un dels problemes més freqüents que poden aparèixer, i afecta directament la qualitat dels resultats obtinguts. Està relacionat amb com s'ajusta la xarxa neuronal al conjunt d'entrenament.

En el cas de l'*underfitting* és el problema més evident, ja que en aquest cas s'observa, per a tots els conjunts un valors de les mètriques molt semblants, un error molt alt i unes prediccions dolentes. Els resultats són una generalització massa elevada, ja que el model és molt simple i no pot reflectir la complexitat de la situació real. L'*overfitting*, però, no és tan obvi, i està causat per un model excessivament complex en relació a les dades. En aquest cas, mentre s'observa un error al conjunt d'entrenament que va disminuint, aquest augmenta progressivament als conjunts de validació i de test. Per tant, la xarxa ajusta els pesos únicament en funció de les dades del conjunt d'entrenament, adaptant-se al soroll i als valors particulars d'aquest d'una forma molt detallada, però és incapaç de realitzar una predicció acurada per a unes dades diferents, al intentar reproduir els patrons apresos.

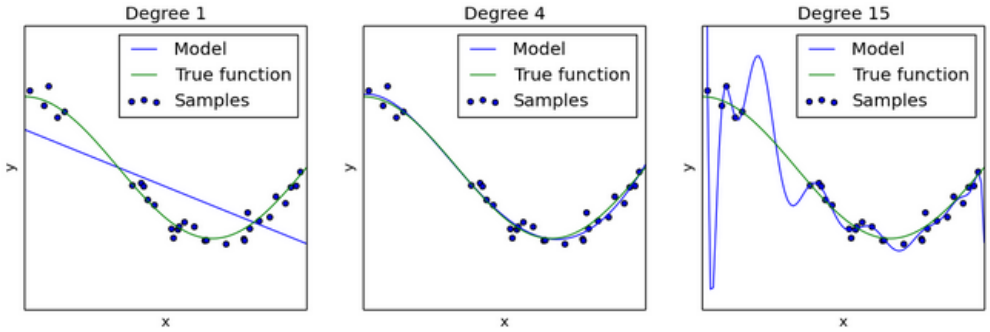

*Figura 2.3. Diferents models. D'esquerra a dreta: cas d'*underfitting*, cas amb un ajust correcte i cas d'*overfitting*.*
Font: [6]

En la Figura 2.3 es pot observar quines implicacions té l'*underfitting* i l'*overfitting* d'una forma senzilla, relacionant el grau de la funció real amb el del model. Pel primer cas, es pot observar com el model realitza una aproximació a la funció real mitjançant una recta (polinomi de grau 1). Al segon cas es veu com és un model amb un bon ajust, on el grau del polinomi és coincident amb el de la funció real. Al tercer i darrer cas es veuen les coincidències del *overfitting* i, per tant, amb el fet de tenir un model altament complex: s'ajusta únicament a les mostres del conjunt d'entrenament amb un polinomi de grau 15, però a la realitat aquestes tendències seran irreals, donant lloc a una predicció errònia. Per tant, una dels principals aspectes a controlar i a determinar durant el treball serà l'assoliment d'un model que proporcioni el millor ajust possible i evitant els dos problemes principals esmentats en aquest capítol.

3. Context històric

3.1. Història de la intel·ligència artificial

En aquest Treball Fi de Grau es desenvoluparà una xarxa neuronal amb l'objectiu de realitzar prediccions sobre consum energètic. Les xarxes neuronals són els models amb els quals es desenvolupa el *Deep Learning*, que a la seva vegada és un subcamp del que es coneix com a *Machine Learning* que forma part de les diferents tècniques que integren el camp de la intel•ligència artificial.

La intel•ligència artificial va néixer a la dècada de 1950, però es va plantejar molt abans. El 1842, la matemàtica britànica Ada Lovelace va plantejar la possibilitat de desenvolupar la idea de què les computadores poguessin arribar a pensar i a aprendre per elles mateixes, però la va refusar [7][8]. Aquestes primeres hipòtesis van ser recollides, molts anys després, el 1950, pel matemàtic Alan Turing a l'article *Computing Machinery and Intelligence*, al qual, a més d'introduir el famós test de Turing, es van establir les bases teòriques de la intel•ligència artificial (tot i que en aquell moment no es coneixia per aquest nom) i del *Machine Learning*, tot contradint la visió d'Ada Lovelace i considerant la possibilitat de què les màquines pensin i aprenguin per elles mateixes, fins i tot planificant un programa de recerca de 50 anys durant els quals s'implementarien i es desenvoluparien totes les teories sobre aquests àmbits [9].

Es considerà però, que l'inici del desenvolupament de la intel•ligència artificial va succeir el 1956 durant la *Dartmouth Summer Research Project on Artificial Intelligence* (DSRPAI). Va ser la primera vegada que es va emprar aquest terme, *Artificial Intelligence* per John McCarthy, també inventor de Lisp, el primer llenguatge de programació utilitzats en aquest camp, implementat entre 1958 i 1962 [10]. Va ser el congrés on es va presentar el que es considera com el primer programa d'intel•ligència artificial, *Logic theorist*, creat per Allen Newell, Cliff Shaw i Herbert Simon. Aquest programa tenia la funció d'imitar l'habilitat dels éssers humans en la resolució de problemes. A partir d'aquest moment, i en el període comprès entre 1957 i 1974, es produeix un gran avenç i investigació en aquest camp, amb una popularització del *Machine Learning* i el desenvolupament del *General Problem Solver* de Allen Newell i Herbert Simon enfocat a la resolució de problemes i ELIZA, creat per Joseph Weizenbaum i amb l'objectiu de interpretar el llenguatge oral. En aquests primers

moments, les investigacions es basaven en aplicar les idees de la lògica deductiva, coneguts com a demostració de teoremes, a problemes de raonament humà, creant axiomes lògics per solucionar els problemes. Aquest enfoc, conegut com a intel•ligència artificial simbòlica, plantejava un gran problema, ja que no es considerava la possibilitat de la incertesa i, per tant, amb el pas del temps aquesta idea s'ha descartat.

Tota aquesta investigació, a més de les inversions que van realitzar els governs dels Estats Units i Regne Unit van provocar que hi hagués una gran esperança i optimisme al voltant de totes aquestes tècniques, potser una reacció massa eufòrica que va topar amb alguns dels problemes que encara presentaven aquests mètodes, entre ells, com un dels més importants, les limitacions en la capacitat de les computadores de la època, que van fer que gran part de les companyies i governs que van invertir en aquestes investigacions deixessin de fer-ho, el que va provocar que s'entrés en un període conegut com l'hivern de la intel•ligència artificial (*Winter AI*). Degut a això, entre 1974 i 1980 no es va realitzar cap avenç important [11] [12].

A l'inici de la dècada de 1980, es van produir una sèrie de factors que van provocar que l'interès per la intel•ligència artificial es tornés a reactivar: va aparèixer una nova forma de les intel•ligències artificials simbòliques, coneguda com a *expert systems* (utilitzant regles derivades dels coneixements experts) i introduïdes per Edward Feigenbaum que es van començar a popularitzar a l'àmbit de les grans multinacionals, la popularització de les tècniques de *deep learning* de la mà de John Hopfield i David Rumelhart i una reactivació de les inversions que es van produir anteriorment, especialment de mà del govern japonès, a part dels d'Estats Units i del Regne Unit, amb 400 milions de dòlars en el període comprès entre 1982 i 1990 dins del programa conegut com a *Fifth Generation Computer Project*. Aquestes inversions també es van produir a càrrec de les pròpies empreses: es calcula que, a l'any 1985, les companyies estaven invertint aproximadament 1 bilió de dòlars a l'any en aquesta tecnologia.

Malgrat tot aquests canvis, aproximadament al 1987, es van començar a abandonar, una altra vegada, aquestes tècniques, i principalment per motius semblants als que havien provocat el primer hivern de la intel•ligència artificial: la escassa potència dels ordinadors de la època (tot i la gran millora que havien experimentat des de la dècada de 1960), i els elevats costos que comportava. També va contribuir a això un dels principals problemes que presentaven aquests sistemes basats en regles: no eren capaços d'expressar teories coherents per problemes als quals hi havia un cert grau d'incertesa.

A més, va haver-hi un altre factor que va ser determinant en el seu abandonament temporal: l'aparició i popularització dels equips informàtics amb els sistemes operatius de Windows i Apple, molt més potents i econòmiques que les màquines que funcionaven amb Lisp, el llenguatge de programació que encara s'emprava de forma majoritària per a la intel•ligència artificial, van fer que les empreses es decantessin pels primers. Tots aquests factors van donar lloc al període que es coneix com el segon hivern de la intel•ligència artificial (*Second Winter AI*) entre 1987 i 1993.

A partir de 1993, una sèrie d'esdeveniments va propiciar un ressorgiment de les tècniques relacionades amb la intel•ligència artificial. Primer de tot, els ordinadors van adquirir una potència ja suficient per desenvolupar els processos, i un nou enfoc basat en la resolució de problemes més específics i en l'ús de tècniques i mètodes probabilístics i estadístics per lidiar amb situacions d'incertesa, especialment al camp del *machine learning*.

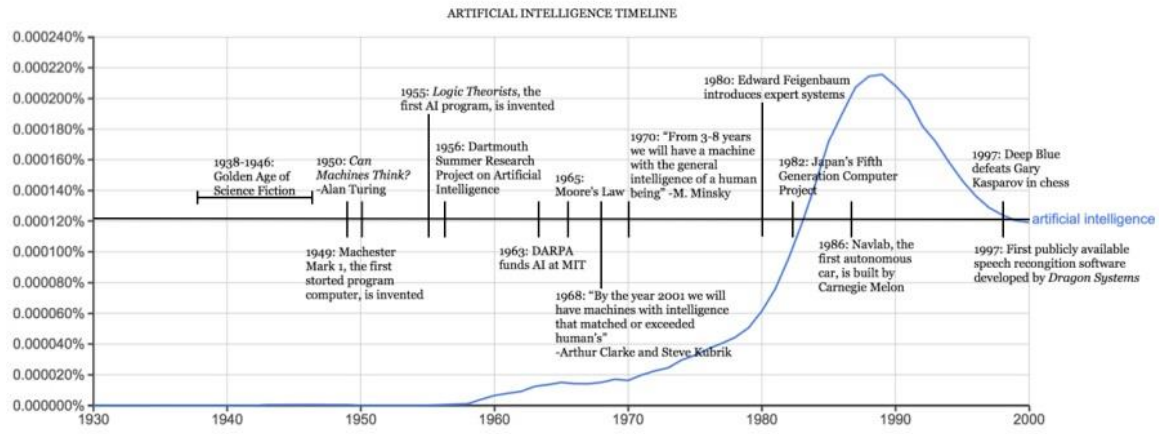

Figura 3.1. Línia temporal de l'evolució de la intel·ligència artificial. Font: [11]

3.2. Història de les xarxes neuronals

Si centrem aquesta visió històrica sobre la història de les xarxes neuronals, el tema sobre el qual versa aquest Treball Fi de Grau, les seves fites més destacables són les següents: el seu origen es remunta a l'any 1943, quan el neuropsicòleg Warren McCulloch i el matemàtic Walter Pitts, per descriure com funciona un cervell humà, van emprar un circuit elèctric per simular el seu funcionament, i van crear el que es coneix com a model neuronal de McCulloch-Pitts. Van descobrir que aquestes neurones poden rebre impulsos tant negatius com postitius que les fan entrar en funcionament i que en el fet de què una neurona

"s'encengui" es realitza una computació en funció de la sinapsi (excitatòria o inhibitòria) que s'hagi produït (la sinapsi és la connexió entre neurones), per tant, és possible determinar matemàticament els efectes dels diferents estímuls en cada neurona. El funcionament d'una neurona McCulloch-Pitts es pot descriure de la següent forma: durant un temps, si no arriba una entrada inhibitòria es sumen les entrades d'excitació, es comprova si arriben al llindar que provoquen l'encesa de la neurona i actua en conseqüència.

Actualment, segons aquest model de neurones, el que dona la capacitat d'adaptació prové del que es coneix com a pesos, que són un valor que s'assigna a cadascuna de les connexions entre neurones (sinapsis), indicant la contribució que ha de tenir cada entrada al fet de què es superi el llindar que marca l'activació de la neurona. Els pesos positius representen sinapsis excitatòries i els negatius sinapsis inhibitòries. L'encesa de la neurona està representada per un 1 i l'apagada per un 0. Es pot representar l'equació que governa l'encesa de la neurona com:

$$\sum_{i=1}^{n} X_i \times W_i + b \geq 0$$

(Equació 3.1)

On W_i representa el pes de cadascuna de les *i* entrades, X_i és la informació de la entrada *i* i b és un terme de biaix que ajuda a la neurona a determinar quan s'arriba al llindar que marca la seva encesa [13].

Més tard, al 1949, Donald Hebb va escriure *The Organitzation of Behaviour,* on l'autor suggereix que les connexions entre neurones es veuen reforçades cada vegada que s'utilitzen, el que indicaria que amb un entrenament iteratiu la connexió millora i l'aprenentatge seria millor. No va ser fins als anys 50, però, quan es va poder simular el funcionament d'aquestes xarxes neuronals. Els primers intents, tot i que infructuosos, es van portar a terme als laboratoris d'IBM [18].

No va ser fins el 1958 quan Frank Rosenblatt va proposar l'estructura del perceptró, que representa la base de totes les teories sobre xarxes neuronals proposades a partir d'aquell moment. Representa una variant de les neurones proposades per McCulloch-Pitts, i les seves principals diferències són les següents: els pesos i els llindars no són tots iguals, els

pesos de les neurones poden ser tant positius com negatius, no hi ha una sinapsi inhibitòria específica, la sortida pot prendre com a valors 1 o -1 (en comptes de 1 o 0 al cas de la neurona de McCulloch-Pitts) i tenen una regla d'aprenentatge. Aquesta darrera diferència és la més important, ja que significa que el perceptró pot aprendre i per tant, reproduir un patró o comportament a partir de les entrades que rebi. L'estructura d'aquest perceptró primigeni es pot dividir en tres capes (tot i que aquest concepte, capes, no té relació amb un perceptró multicapa) [14]:

- La primera està formada pel que es coneix com la *retina* del perceptró (*S-Units).*
- La segona capa conté les unitats d'associació, que realitza la suma ponderada d'un conjunt de neurones de la capa anterior seleccionat de forma aleatòria. Els seus pesos estan fixat, no té la capacitat d'aprendre (*A-Units).*
- La tercera capa conté les unitats de resposta, i cadascuna d'elles està connectada a un conjunt concret d'unitats d'associació. Si la suma de les entrades ponderades és més gran que el llindar, la sortida pren el valor +1, en cas contrari, el seu valor és -1. Els pesos d'aquestes unitats sí que son ajustables, i són els que li donen a la neurona la capacitat d'aprendre *(R-Units).*

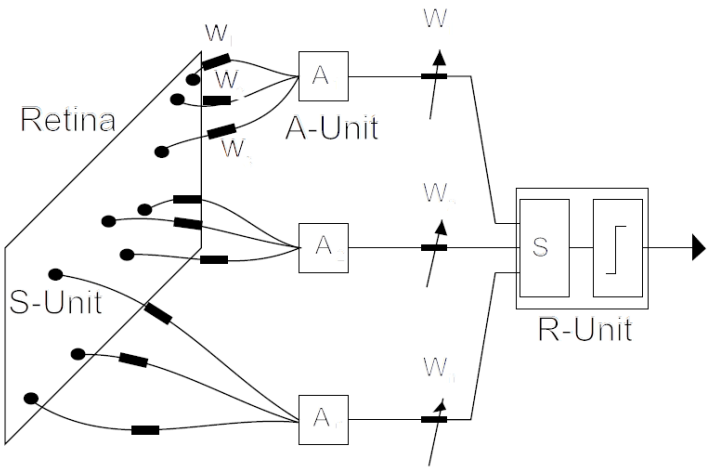

Figura 3.2. Estructura del perceptró de Rosenblatt. Font:[15]

Per realitzar l'aprenentatge, es realitza el següent mètode: si la sortida que es proporciona és correcta, no es realitza cap canvi als pesos de la darrera capa, i si és incorrecta hi ha dues possibles solucions: o bé disminuir els pesos de les neurones actives i augmentar el llindar si aquestes s'activen quan no ho han de fer, o bé incrementar els pesos i reduir el llindar si la neurona no s'encén quan ho hauria de fer.

Un any més tard, al 1959, Bernard Widrow i Marcian Hoff van desenvolupar els models coneguts com a ADALINE (*Adaptative Linear Elements*) i MADALINE (*Multiple Adaptative Linear Elements*). Aquesta darrera va suposar la primera aplicació d'una xarxa neuronal a un problema del món real, ja que va ser creada per ajudar a eliminar el ressò de les trucades de telèfon.

Tot i aquestes primeres investigacions i aplicacions, a final de la dècada de 1960 aquest enfoc va caure en desús, principalment degut a la visió crítica de les mateixes que s'oferia al llibre de 1969 escrit per Marvin Minsky i Seymour Papert *Perceptrons,* al qual es destacaven els principals problemes dels perceptrons d'una única capa, la seva incapacitat per calcular funcions que no són linealment separables. Una de les limitacions al seu desenvolupament va ser, tal i com va ocórrer amb la resta de tècniques englobades dins de la intel·ligència artificial, la limitació en la capacitat computacional de la època: això va provocar que en les dècades de 1970 i inicis de la dècada de 1980, les xarxes neuronals també patissin el *Winter AI*.

En aquest període, concretament a l'any 1974, es va produir un gran avenç que en aquell moment va passar pràcticament desapercebut, principalment degut a què fins a 1975 no es va desenvolupar el primer perceptró multicapa: Paul Werbos va proposar l'algorisme d'aprenentatge conegut com a *backpropagation* (retropropagació), i el seu funcionament és el següent: primer es calcula l'activació de cada neurona aplicant l'entrada a la neurona de forma progressiva, des de la primera capa fins la darrera, i posteriorment els pesos s'actualitzen anant des de la capa de sortida fins la d'entrada. Això es realitza calculant el gradient de l'error [16]. També va haver-hi diferents propostes durant aquesta època, principalment de la mà d'Igor Aleksander, que va crear la *Binary Discriminant Neuron,* principalment destinada a problemes on s'havien de realitzar classificacions dins de 2 categories.

A inicis de la dècada de 1980, concretament el 1982, John Hopfield va proposar la creació de xarxes neuronals en les quals les neurones estaven connectades en ambdues direccions, conegudes com a RNN per les sigles en anglès de *Recurrent Neural Network*. Reilly i Cooper van presentar una altra xarxa amb múltiples capes, on cadascuna d'elles seguia una estratègia diferent per solucionar el problema. El 1986 es va reprendre la idea de *backpropagation* proposada 12 anys enrere, i es van desenvolupar models multicapa que feien servir aquesta tècnica. El 1997, es va crear una variant de les xarxes neuronals recurrents, *Long Short Term Memory*, a les quals s'emmagatzema informació de les

entrades anteriors, pel que són molt utilitzades en problemes als què s'ha d'aprendre una seqüència [17]. A més, la seva estructura permet solucionar un dels problemes que presenten les altres estructures de RNN, l'esvaniment del gradient, què succeeix perquè la xarxa no és capaç de recordar seqüències en un espai de temps llarg, el que fa que sigui incapaç d'aprendre.

En la actualitat, aquests models són utilitzats en un gran nombre d'aplicacions, tant per tots aquests descobriments que han fet de les xarxes neuronals un instrument realment útil com per l'increment de potència dels ordinadors que permet entrenar models cada vegada més complexos en un temps relativament curt.

4. Inicialització de la xarxa

4.1. Informació sobre les dades

Les dades de les quals s'ha partit inicialment corresponen al consum elèctric d'un habitatge de la ciutat de Barcelona, recollits amb un període de mostratge d'1 minut des de l'1 de gener de 2015 fins al 31 de maig de 2019. Són unes dades molt importants, ja que la presència de certs elements pot ajudar a establir certs patrons de consum, el que pot ajudar en posteriors passes del treball. Aquestes mesures s'han realitzat mitjançant un sistema *Efergy* amb una precisió, segons especificacions, superior al 90%, i es troben en format CSV. El fitxer conté el dia, mes i any al qual es va realitzar la mesura, a més de a quina hora i minut es va produir i el consum elèctric en Watts per minut. D'aquesta casa coneixem més dades: es tracta d'un pis de 90m^2 útils amb 4 habitants, dos adults i dos nens. Els fogons i la calefacció funcionen amb gas, per tant, el seu ús no té influència a l'estudi. Els aparells elèctrics més destacables que es fan servir a l'habitatge són els següents:

- Dos aires condicionats, ubicats a la zona del menjador i a la zona de les habitacions, del model *Fujitsu* ASYG12LECA.
- Una nevera (*Bosch* combo estàndard)
- Una rentadora, model *Zanussi* ZWG6100.
- Un rentaplats marca *Siemens*.
- Una màquina de cafè expresso.
- Dos routers i un conversor de fibra.

I els hàbits de consum són:

- Es fa una rentadora al dia, a primera hora del matí.
- El cafè es fa amb la màquina expresso.
- S'engega el rentaplats alguns dies de la setmana al vespre.
- L'aire condicionat s'utilitza en mode estàndard a la tarda i en mode *economy* a la nit.

- Els routers i el conversor de fibra estan sempre encesos.

A partir d'aquestes dades, podem extreure certes conclusions: sabem que la calefacció funciona amb gas però que hi ha un sistema d'aire condicionat instal·lat: previsiblement els consums elèctrics seran més alts a l'estiu que a l'hivern. A més, sabent que es posa la rentadora al matí i es fan els cafès a aquesta hora també, és esperable que durant aquest període es produeixi un pic de consum.

4.2. Pre-processat de les dades

Les dades de partida es troben en diferents fitxers: les dades des de l'1 de gener de 2015 fins al 31 de desembre de 2018 es trobaven en un fitxer en format .csv i les dades mensuals des de l'1 de gener de 2019 fins al 31 de maig de 2019 estaven en fitxers .csv independents. Cadascuna d'aquests fitxers tenia una estructura distinta, per tant, la següent tasca va ser unificar totes aquestes dades en un únic arxiu, per tal de poder processar-les posteriorment. La solució trobada va ser la d'unificar manualment els arxius dels darrers 6 mesos en format .txt i després escriure un programa amb Python per tal d'unificar els dos arxius. En aquest pas s'ha afegit un *flag*, en aquest cas, un element binari que indica si la medició de consum corresponent a aquest minut és vàlida o no. Una vegada estan unides totes les dades s'emmagatzemen en un fitxer .txt, que dona la possibilitat de posteriorment aplicar el format adient depenent de les necessitats.

Amb un altre programa es decideix processar les dades guardades a l'arxiu esmentat anteriorment i passar-les a un format adequat per una xarxa neuronal. Per tant, les dades queden emmagatzemades en un format de tensor (també conegut, a partir d'ara, com a *array* o matriu),les dades corresponents a aquell minut i la informació addicional del mateix es codifiquen a un vector, després, tots els vectors que fan referència a minuts d'un mateix dia s'agrupen a una matriu, i totes aquestes matrius formen part d'una matriu més gran. La següent imatge pot ajudar a comprendre com és aquesta estructura:

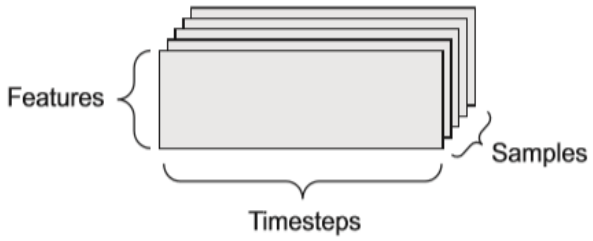

Figura 4.1. Estructura d'un tensor 3D. Font: [19]

Aquesta estructura s'aconsegueix fent servir la llibreria *NumPy*, de la qual els arxius fan servir l'extensió .npy, que guarda els *arrays* en un format que fa més fàcil obrir-lo i utilitzar les dades posteriorment. És imprescindible que les dades estiguin guardades en aquest format a l'hora d'introduir-les a la xarxa neuronal, ja que és l'únic admès per les biblioteques utilitzades per la creació de la xarxa *Keras* i *TensorFlow*. Els principals avantatges que proporciona aquesta forma d'emmagatzemar les dades és que permet realitzar operacions amb vectors i, com que totes les dades que conté són d'un únic tipus, els càlculs es realitzen més ràpidament en comparació amb altres formes d'escriure dades com bé poden ser les llistes, al no haver de guardar informació addicional sobre cada una d'elles [20].

Tot i que als primers estudis només es faran servir les dades temporals que es consideren bàsiques (minut del dia al qual pertany la mesura del consum i el dia respecte l'inici de la presa de dades), es tenen altres dades. Es crea una variable que conté el mes al qual pertany la mesura. El rang de valors d'aquesta variable va des de 1 per al mes de gener i 12 per al mes de desembre. A més, també es guarda a quin dia de la setmana es va registrar el consum, amb l'objectiu de no perdre cap informació que es té de l'arxiu original. Tenir aquesta dada pot ser molt important a l'hora de realitzar la predicció, ja que la demanda d'energia d'un habitatge varia molt segons a quin dia de la setmana ens trobem. Els dies de la setmana també s'han codificat fent servir un número, ja que les xarxes neuronals necessiten treballar amb dades en format numèric. Es fa correspondre el número 1 amb el dilluns i el número 7 amb el diumenge.

La següent passa és estudiar la reducció del nombre de dades de forma que es pugui minimitzar els temps de càlcul sense que es redueixi la precisió. S'ha modificat la freqüència amb la qual estaven preses les dades, ja que el període de mostreig de les mesures inicials era d'1 minut, fet que provocava una elevada inestabilitat. A més, aquest gran nombre de dades pot afegir complexitat innecessària, ja que pot haver minuts on es produeixin consums massa elevats per circumstàncies no previsibles (per exemple, encesa de diferents electrodomèstics alhora). Aquests valors no es poden predir a la xarxa neuronal: per tant, tampoc té gaire sentit mantenir-los, ja que només afegirien soroll a les mostres i fins i tot dificultarien el funcionament de la xarxa. La presència de minuts als quals, bé per una interrupció en el subministrament elèctric o bé per errors al sistema que s'encarrega de captar les dades, no hi ha registre de consum pot suposar un problema. S'ha creat una variable binària, que es fa servir com a *flag* per indicar quines mostres són vàlides (prenen el valor 1) i quines no (prenen el valor 0), i posteriorment realitzar el tractament adient a

aquestes dades.

Hi havia minuts, com s'ha dit anteriorment, que no tenien cap consum registrat, per tant, s'ha realitzat la interpolació dels valors que faltaven dels dies que tenien més de 720 valors de consum registrats (és a dir, més de la meitat) i posteriorment s'han delmat les dades fins a tenir una mostra cada cinc minuts. El procés d'interpolació serveix per a poder tenir una estimació dels valors que falten i que totes els dies considerats tinguin el mateix nombre de dades, fet que possibilita la delmació, ja que amb la funció *decimate* proporcionada per la biblioteca *SciPy Signal* i que realitza aquesta tasca és necessari que totes les mostres tinguin el mateix nombre de dades. Així, s'ha reduït el nombre de dades per dia de 1440 fins a 288. S'ha considerat que amb la meitat de les dades d'un dia ja es pot obtenir una mostra significativa de com és el perfil de consum d'un dia. A més com es pot observar al gràfic anterior, no hi ha molts dies als quals hi hagi una falta absoluta de dades de consum, això es produeix només al període al qual va haver-hi problemes de connexió amb el router i les dades es van perdre. Per això s'ha fet servir com a indicador el *flag* incorporat anteriorment. Aquest procés de delmació s'ha realitzat amb un filtre passa baixos FIR (*finite impulse response*) d'ordre 4, suficient per a atenuar gran part dels valors inusualment elevats que es registraven.

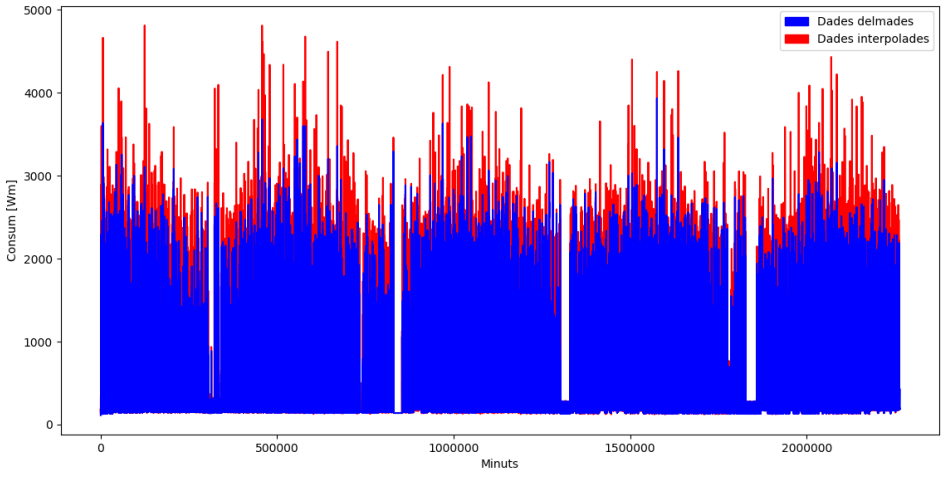

Figura 4.2. Comparativa entre dades delmades i interpolades

Posteriorment, s'han creat els conjunts d'entrenament, validació i test. Per a aquests, s'ha triat una divisió 70-20-10, ja que és important tenir un conjunt de validació relativament gran [21], per això s'ha triat que aquest sigui més nombrós en detriment de la mida del conjunt de test. Pel conjunt d'entrenament, s'ha triat que sigui un 70% del total, una mida sobre la qual

hi ha cert consens.[22][23]. Les mostres de cada conjunt s'han aleatoritzat per tal de no introduir les dades en el mateix ordre al qual s'han pres, fet que podria portar problemes a l'hora d'entrenar la xarxa neuronal, especialment si es fa servir, com s'explicarà més endavant, l'optimitzador SGD [24] ja que podria succeir que la xarxa aprengués l'ordre de les dades únicament i nos fos capaç d'ajustar-se realment als paràmetres introduïts, ja que tindria un biaix causat per aquest ordre. Per decidir les dades de quant de temps es necessiten per poder predir el consum d'un dia s'ha escrit un programa amb Python per a realitzar un estudi de l'autocorrelació. Als següents gràfics s'observen els resultats:

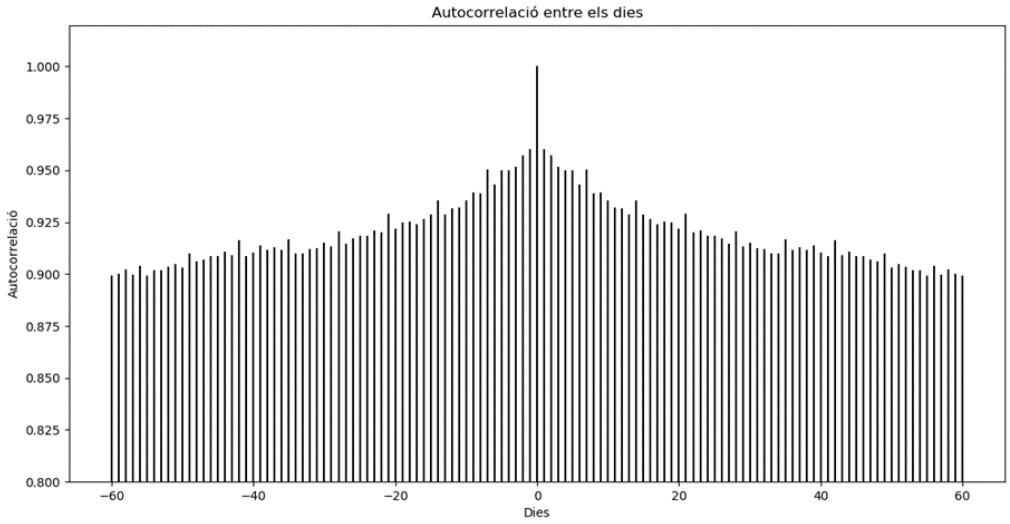

Figura 4.3. Gràfic d'autocorrelació entre els dies

En el gràfic d'autocorrelació entre els dies, podem observar que hi ha certa periodicitat setmanal, ja que el coeficient d'autocorrelació presenta una lleugera pujada cada 7 dies.

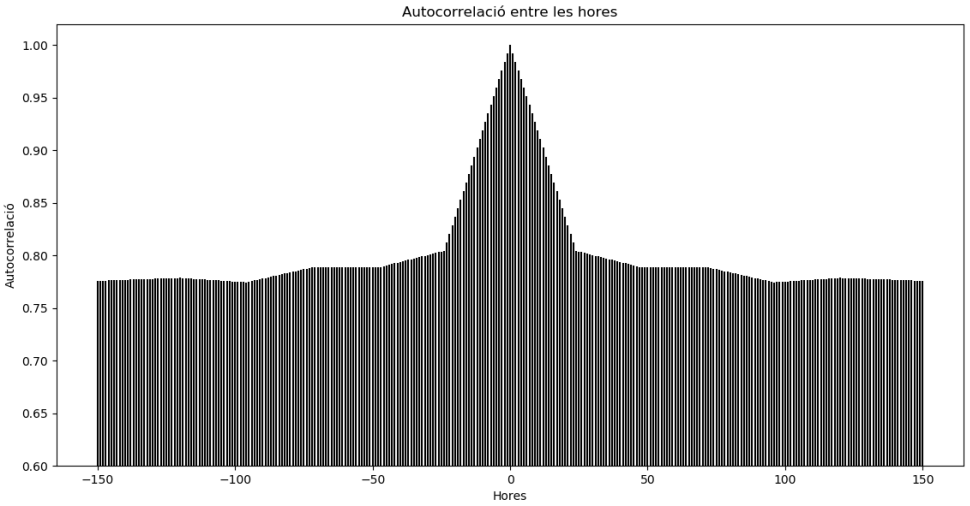

Figura 4.4. Gràfic d'autocorrelació entre les hores

En el gràfic de correlació entre les hores, es pot veure que aquesta disminueix de forma significativa passades 24 hores i es manté més estable a partir d'aquestes.

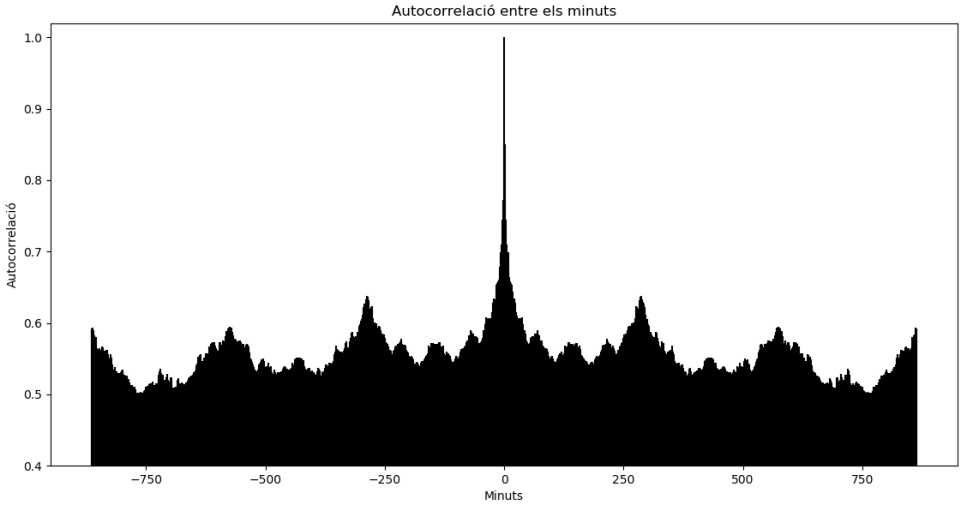

Figura 4.5. Gràfic d'autocorrelació entre els minuts

Si s'estudia l'autocorrelació entre els minuts, es pot extreure informació semblant a quan s'estudia la informació de les hores: les dades tenen una periodicitat més marcada cada 24 h (equivalent a 288 minuts). També es pot observar un coeficient d'autocorrelació més elevat cada 48 h (576 minuts), però aquest no sembla tan important. Per tant, per a les proves s'alimentarà la xarxa neuronal amb les dades d'un període de 24 hores i es prediran les de les 24 hores següents.

Un altre dels factors importants que pot influir en el consum elèctric d'un habitatge és la climatologia de la zona a la qual es troba. Amb la finalitat de comptar amb aquesta informació, es van descarregar de la pàgina web de l'AEMET (Agència Estatal de Meteorologia) les dades corresponents als dies dels quatre anys i mig durant els què es va portar a terme la recollida de les dades de consum. L'arxiu obtingut es troba en format *json,* per a llegir les dades s'ha utilitzat el mòdul de Python que porta el mateix nom. S'ha seleccionat l'estació amb codi 0201D per ser la més propera a la zona de Sagrada Família, on es troba ubicat l'habitatge. De totes les dades de les què es disposa de cada dia, s'han triat les de la temperatura màxima, mínima i mitjana de cada dia, perquè són les que més relació tenen amb el consum elèctric, al influir en la demanda que es fa d'equips com l'aire condicionat.

4.3. Creació del codi amb Python

Una vegada es tenen les dades en un format adient, es crea el codi de la xarxa neuronal, Per aquesta passa han estat imprescindibles dos mòduls: *Keras,* que és una biblioteca *open source* especialment dissenyada per a la creació de xarxes neuronals per François Chollet, enginyer de Google. És capaç d'executar-se amb els *backends TensorFlow, CNTK* o *Theano*. De totes les opcions possibles, s'ha triat fer servir *Keras* perquè proporciona una experiència simple i ràpida per l'usuari, a més de tenir les accions bàsiques de configuració de la xarxa neuronal ja creades, i els resultats són de la mateixa qualitat que en programes d'ús molt més complex. S'ha triat *TensorFlow* com a *backend* per ser el que té una integració més natural amb *Keras,* a més de tenir una major comunitat d'usuaris que, en cas de tenir un problema ofereixen un suport superior a la resta de biblioteques,

El primer pas, previ a la creació de la xarxa en sí mateixa, és, com s'ha explicat anteriorment, preparar les dades i posar-les a un format concret que la xarxa pugui tractar. La gran part de l'etapa de pre-processat de les dades es realitza amb arxius independents al que conté la xarxa neuronal i es van guardant en fitxers intermedis Una vegada fet això, es crea la xarxa neuronal. Per realitzar-ho s'ha d'escriure el codi en un fitxer de Python amb extensió *.py*. Primer de tot s'ha de triar el tipus de model. En aquest cas s'ha elegit el conegut com a *Sequential*, equivalent a la creació d'una xarxa completament connectada, en la qual cada node d'una capa està connectat amb tots els nodes de la capa següent, és a dir, és el funcionament del que es coneix com a perceptró multicapa, o també en anglès com *feedforward neural networks.* El següent pas és afegir les capes que es vulguin incorporar al

model. La capa d'entrada a la xarxa neuronal ve inclosa per defecte al model i no cal especificar-la. Per a les capes ocultes i la capa de sortida s'ha especificar el tipus que es vol seleccionar: al tractar-se d'un perceptró multicapa, s'han de seleccionar les capes *Dense,* les adequades per aquesta configuració de xarxa neuronal. A cada capa, els paràmetres bàsics que s'han de seleccionar són els següents:

- Nombre de neurones de la capa: a les capes ocultes es pot triar qualsevol nombre: a les capes de sortida aquest nombre ha de coincidir amb el de variables de cadascuna de les files de la matriu de *target.*

- Funció d'activació de cada una de les capes.

- A la primera de les capes ocultes, tot i que la capa d'entrada es crea automàticament, s'especifica la mida del vector d'entrada a la xarxa neuronal.

A continuació, s'ha de compilar el model amb la funció *compile.* És el primer pas per a configurar el model per a l'entrenament. Els paràmetres més importants que es defineixen en aquesta etapa són l'optimitzador, la *loss function* i la mètrica que s'utilitza per mesurar la precisió de la xarxa.

A continuació, si es vol fer que el procés iteratiu finalitzi quan no hi ha cap millora en la mètrica o a la *loss function,* es realitza amb la funció *EarlyStopping,* a la qual s'especifica quin és l'indicador a monitoritzar per controlar la fi de l'entrenament (per defecte, el valor de *loss function* al conjunt de validació), si s'han de restaurar els pesos de la millor iteració (*restore_best_*weights) i quant s'ha d'esperar per observar que no s'ha produït cap canvi positiu en el valor a controlar (*patience)*. L'ús d'aquesta funció també serveix per a evitar el problema de *l'overfitting,* ja que una de les conseqüències que té aquest és que la *loss* del conjunt de validació empitjora mentre que la del conjunt de validació, al aprendre únicament les dades i no realitzar una generalització, continua millorant.

Aquestes condicions s'han de definir de forma prèvia a la utilització de la funció *fit,* que entrena el model per a un determinat nombre d'iteracions. Aquí es defineixen, a més de quantes iteracions es volen realitzar, quins són les dades d'entrada i les dades objectiu (és a dir, les matrius d'*input* i *target corresponents a l'entrenament*) i la mida de lot. Aquesta variable indica que, dins de cada iteració, s'agafa un determinat nombre de dades, es fa el càlcul, s'ajusten els pesos i repeteix la mateixa operació amb unes altres dades que no s'havien fet servir fins aquell moment, fins que les utilitza totes i passa a la següent iteració.

A més, la funció *fit* serveix per incorporar al model altres paràmetres mitjançant l'argument *callbacks,* com per exemple la funció *EarlyStopping* explicada anteriorment.

Definint totes aquestes funcions i amb l'estructura anterior, el procés d'entrenament ja estaria definit: no obstant això, encara falta alguna eina que permeti visualitzar tant les prediccions com els resultats. Amb la funció *predict* es pot generar una predicció, fent servir la xarxa neuronal definida anteriorment, per a un conjunt de dades, en aquest cas, s'ha fet servir per obtenir els resultats per al conjunt de test. Després, per visualitzar els valors de *loss* i de la mètrica utilitzada s'utilitza la funció *evaluate,* que els calcula indicant quins són les matrius d'*input* i *target* del conjunt de test. Amb tot l'anterior ja s'obtenen els valors numèrics necessaris per a realitzar una avaluació de la xarxa neuronal, però encara no es mostra cap gràfic, el que fa que la interpretació de les dades no sigui gens intuïtiva. A continuació es mostra quin és el codi per a una de les proves, concretament la 2.3.7, on es poden observar totes les parts detallades anteriorment.

```
model = Sequential()
model.add(layers.Dense(350,input_shape=(579,),activation="relu"))
model.add(layers.Dense(350,activation="relu"))
model.add(layers.Dense(288,activation="sigmoid"))

model.compile(optimizer=adam(), loss="mse", metrics=[tf.keras.metrics.RootMeanSquaredError(name="rmse")])
acaba_abans=EarlyStopping(patience=10,restore_best_weights=True)
history=model.fit(inputs_train,targets_train,epochs=550, batch_size=128, validation_data=(inputs_val, targets_val),callbacks=[acaba_abans])
predits=model.predict(inputs_test)
test_score = model.evaluate(inputs_test, targets_test, verbose=0)
print('test loss, test acc:', test_score)
```

Figura 4.6. Codi de la prova 2.3.7

Per fer més intuïtiu el posterior anàlisi, es realitza un post-processat de les dades i dels resultats. Utilitzant la biblioteca *matplotlib* de Python, es creen els diferents gràfics per a cadascuna de les variables d'interès per a l'estudi. Per a totes les proves que es realitzin, es generaran els següents gràfics, el contingut dels quals es mostra a continuació:

- Primer de tot, es visualitza el gràfic que indica com evoluciona la *loss function* pels conjunts d'entrenament i de validació.
- Després, es mostra el gràfic amb la mètrica que avalua la precisió del model per les dades d'entrenament i de validació.
- En tercer lloc, apareix una taula els valors de *loss* i de la mètrica al conjunt de *test* i el nombre d'iteracions concret al qual s'arriba a la menor *loss* del conjunt de validació (o bé quan es deixa de produir una millora apreciable i significativa).
- El gràfic amb totes les prediccions realitzades al conjunt de *test* superposades amb els valors reals.

- Per últim, i per facilitar la comprensió del gràfic anterior, es visualitzen les prediccions de 6 dies seleccionats de forma aleatòria.

5. Predicció de consum

5.1. Prova 1: Actualització diària de la predicció

Per començar a dissenyar la xarxa neuronal, s'ha realitzat una prova més senzilla consistent en alimentar la xarxa amb el consum del dia anterior i fer que predigués el consum del dia següent. Aquesta estructura seria vàlida en el cas de què es volgués tenir un model que només actualitzés la predicció una vegada al dia. Únicament s'inclouran dos indicadors temporals: el minut respecte a l'inici del dia i el dia respecte a l'inici de la presa de dades. Per tant, es començarà l'estudi alimentant la xarxa neuronal amb les dades de consum del dia anterior, al minut al qual s'han realitzat les mesures, el dia i el minut al qual correspon la mostra que s'ha de predir. Així, les matrius d'*input* i *target* tindran la següent forma:

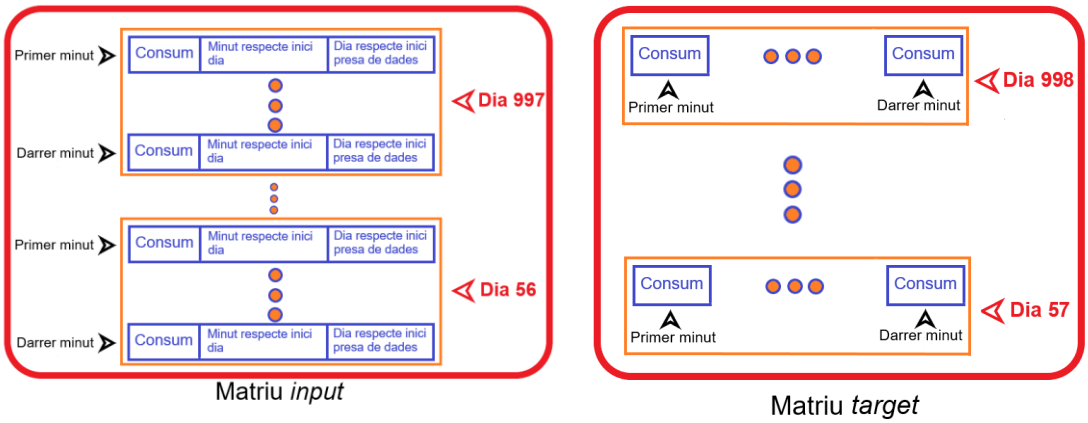

Figura 5.1. Matrius d'input i target

Per això, cal canviar el format de les dades, ja que cadascuna de les característiques té un rang de valors molt diferent (per exemple, els minuts varien des de 5 fins a 1440, mentre que les dades de consum poden anar des de, aproximadament 100 Wm fins a 4000Wm. Això planteja un problema sobre quina forma de normalitzar les dades és la més adient per al problema, ja que és un procediment que es pot fer de moltes formes diferents [25] per això s'han triat dos de les més comuns: una d'elles és el que es coneix com a reescalar, és a dir, calcular, de cada una de les columnes (*features*) del tensor, el màxim i el mínim, i restar a cadascun dels valors individuals el mínim i dividir-lo per la diferència entre el màxim i el mínim. Així, es porten tots els valors al rang comprès entre 0 i 1.

Una altra forma de realitzar-lo, és fer que totes les dades tinguin una mitjana centrada en 0 i

una desviació estàndard d'1, és a dir, normalitzant les dades. Això s'aconsegueix restant a cada valor la mitjana i dividint per la desviació estàndard. Únicament s'ha utilitzat una capa oculta amb 150 neurones i que fa servir la funció d'activació ReLU (*Rectified Linear Unit*), i a la capa de sortida no es fa servir cap funció d'activació, ja que, per a problemes com aquests que no són problemes de classificació i que es tracta de predir un valor més o menys aproximat a la realitat.

La primera prova ha consistit en posar a prova la mateixa xarxa neuronal amb les dues amb les dades normalitzades i reescalades, per veure quin dels dos aconsegueix millors resultats i així fer servir aquest mètode per a la resta d'anàlisis que es realitzin. Com a *loss function* s'ha utilitzat MSE (*Mean Squared Error*), i com a mètrica per a valorar la precisió dels resultats, en un primer moment es va barallar la possibilitat de fer servir la coneguda com a MAPE (*Mean Absolute Percentage Error*), que, al ser un indicador en forma de percentatge, no depèn dels valors concrets resultants de la xarxa neuronal, que a la seva vegada depenen de quin procés s'ha seguit per convertir les dades a un format adient per ser tractades, així, es pot establir una comparació entre els diferents formes de pre-processar les dades, a més de ser molt visual a l'hora d'interpretar les dades. És molt utilitzat per aquests menesters [26], però pot presentar problemes quan hi ha valors propers a zero o que siguin directament zero (el que succeeix en vista a la forma en la qual s'han pre-processat les dades; per tant es farà servir com a indicador MAE (*Mean Absolute Error*) que, al no ser un percentatge, no presenta problemes als valors que són zero o propers a zero i és adient per aquest tipus de dades. Una altra opció que s'ha tingut en compte és la de fer servir RMSE (*Root Mean Squared Error*), i definitivament s'ha utilitzat aquest darrer perquè té en compte la mesura de l'error, és a dir, penalitza més els errors grans que els errors petits i no tots per igual, que és tal i com passaria si es fes servir MAE [27].

Com a optimitzador per a la xarxa neuronal: es farà una comparació entre dos dels més utilitzats en l'actualitat per aquest tipus de problemes: el primer és un dels més típics i coneguts, el gradient de descens estocàstic, més conegut per les seves sigles en anglès SGD (*Stochastic Gradient* Descent). Aquest mètode es basa en realitzar, a cada iteració, un ajustament dels pesos en la direcció de la baixada de l'error més pronunciada, és a dir, la del gradient negatiu) [28]. Tot i que és un procediment que sol portar a solucions acurades, els seus principals inconvenients són la seva lentitud (ja que presenta una taxa d'aprenentatge constant) i la necessitat d'ajustar de forma precisa alguns dels seus paràmetres per assegurar un bon funcionament de la xarxa. L'altre optimitzador que es té en

compte és Adam (*Adaptive Moment Estimation*): un mètode que es va presentar al 2015 i que ha esdevingut molt utilitzat, un dels seus principals avantatges és que rarament necessita que es facin canvis als paràmetres predeterminats, a més, en paraules dels investigadors que ho van introduir "*The method is straightforward to implement, is computationally efficient,has little memory requirements, is invariant to diagonal rescaling of the gradients,and is well suited for problems that are large in terms of data and/or parameters.*" [29]. Així doncs, es provarà quin dels dos optimitzadors proporciona els millors resultats tant pel cas de les dades normalitzades com pel cas de les dades reescalades.

En el cas d'aquesta primera prova, es declara que l'entrenament ha acabat quan, durant tres etapes, ha empitjorat l'indicador de *loss* en el conjunt de validació, o bé quan, si bé la predicció continua millorant, el canvi que experimenta és tant petit que no compensa en front del cost computacional que comporta. Sempre es restauren (i s'utilitzen per avaluar els resultats al conjunt de test) els pesos de les neurones equivalents a la menor *loss* al conjunt de validació.

5.1.1. Prova amb dades normalitzades i optimitzador SGD

Contingut matriu *input*	Consum de cada minut, minut, dia anterior
Contingut matriu *target*	Consum de cada minut del dia següent
Optimitzador	SGD
Mida del lot	32 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1	150
Funció d'activació a la *output layer*	Cap

Taula 5.1. Resum de les característiques de la prova 1.1

A continuació es mostren els gràfics resultants de l'anàlisi:

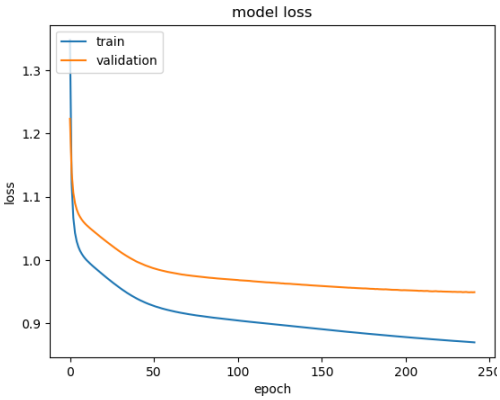

Figura 5.2. Loss de la prova 1.1

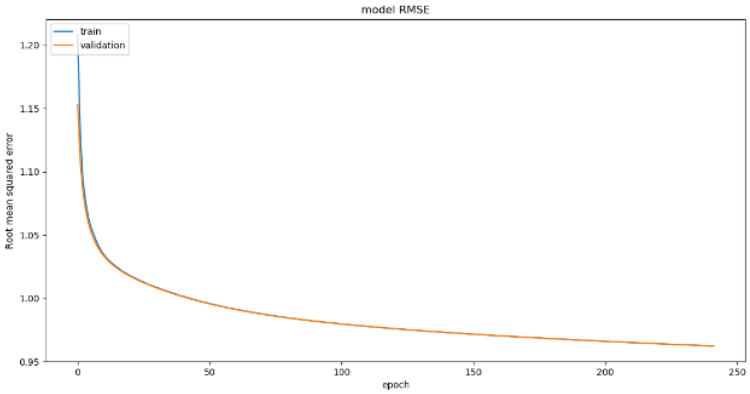

Figura 5.3. RMSE de la prova 1.1

Test *loss*	0.7639
Test RMSE	0.9639
Nombre d'iteracions	239

Taula 5.2. Dades sobre el conjunt de test

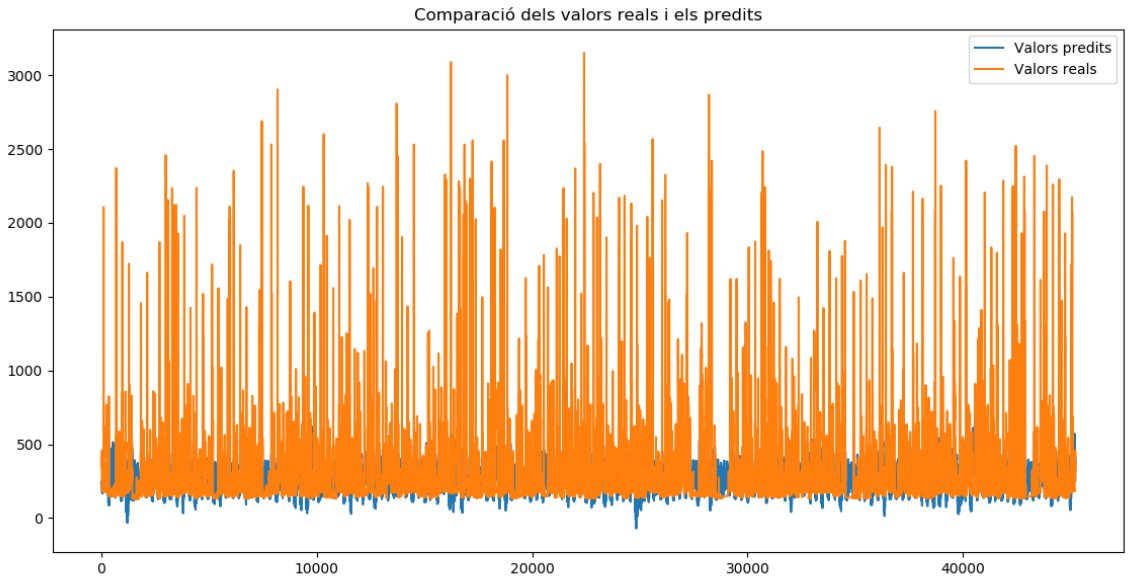

Figura 5.4. Prediccions de la prova 1.1

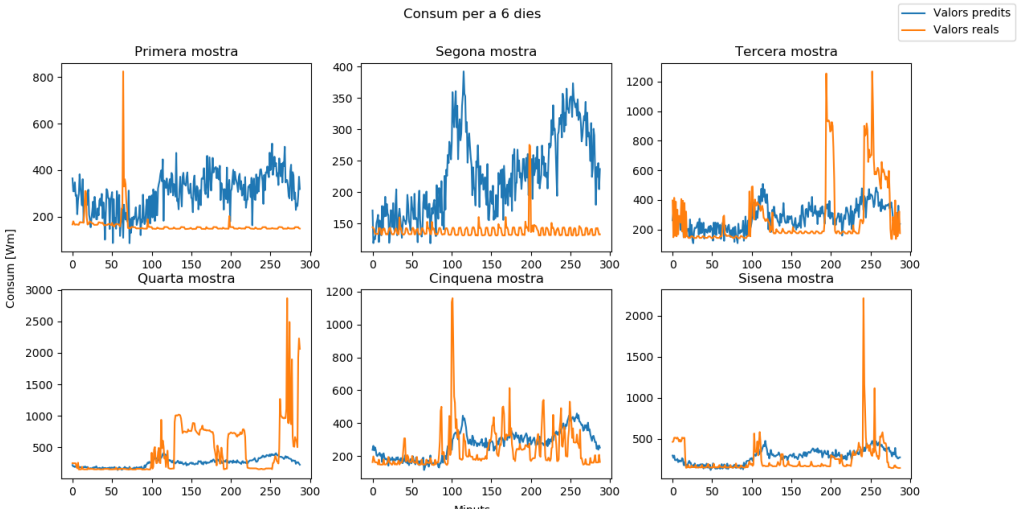

Figura 5.5. 6 mostres de la prova 1.1

5.1.2. Prova amb dades normalitzades i optimizador Adam

Contingut matriu *input*	Consum de cada minut, minut, dia anterior
Contingut matriu *target*	Consum de cada minut del dia següent
Optimitzador	Adam
Mida del lot	32 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1	150
Funció d'activació a la *output layer*	Cap

Taula 5.3. Resum de les característiques de la prova 1.2

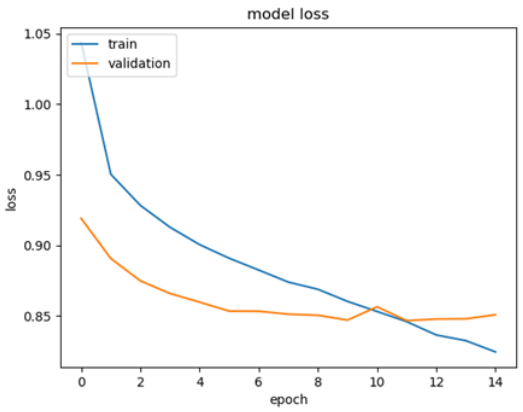

Figura 5.6. Loss de la prova 1.2

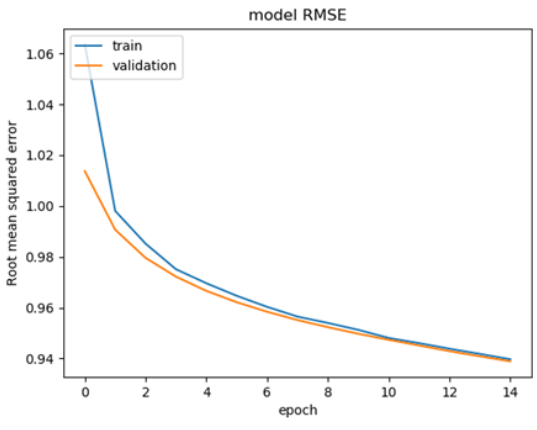

Figura 5.7. RMSE de la prova.1.2

Test *loss*	0.8759
Test RMSE	0.9283
Nombre d'iteracions	11

Taula 5.4. Dades sobre el conjunt de test

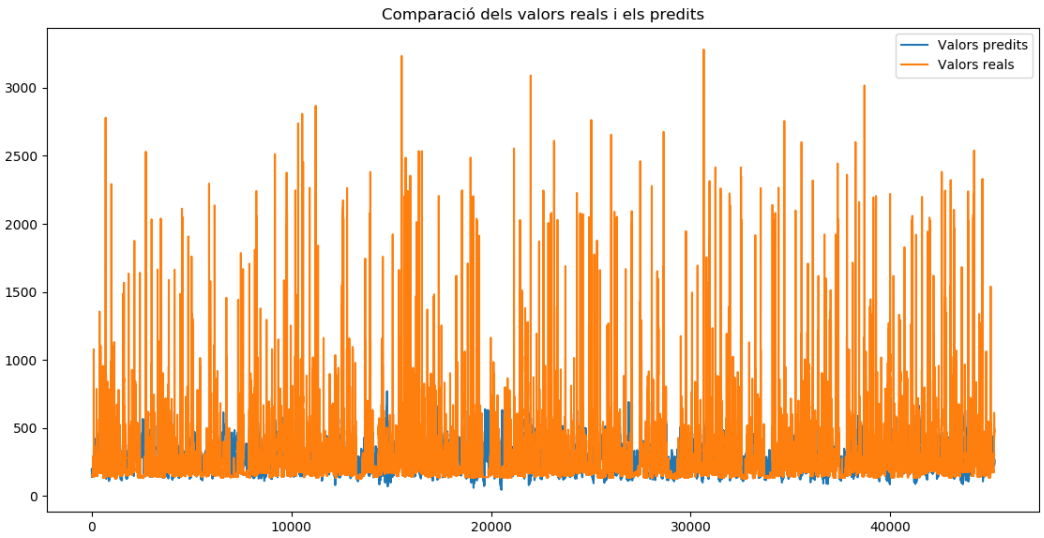

Figura 5.8. Prediccions de la prova 1.2

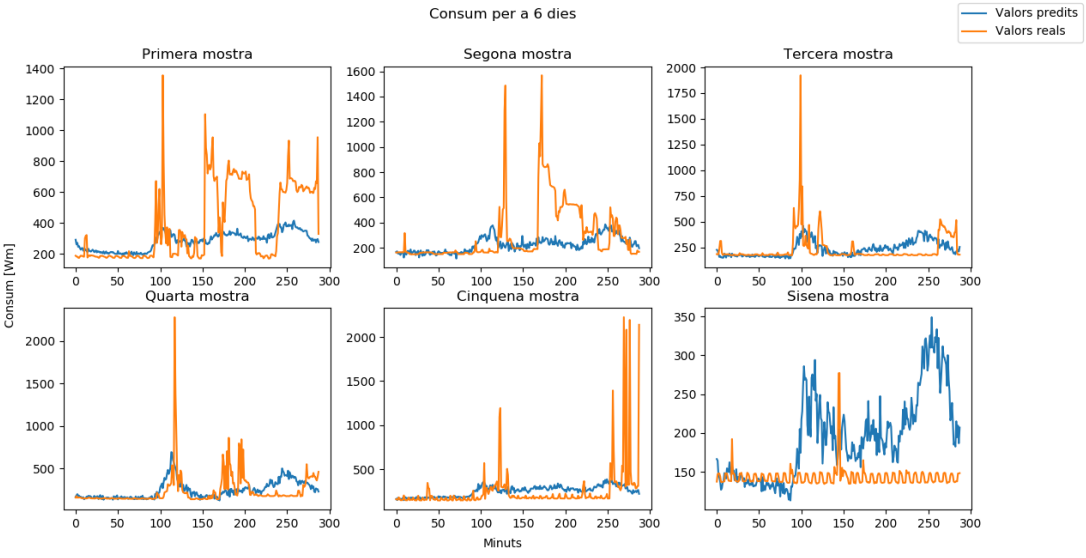

Figura 5.9. 6 mostres de la prova 1.2

5.1.3. Prova amb dades reescalades i optimitzador SGD

Contingut matriu *input*	Consum de cada minut, minut, dia anterior
Contingut matriu *target*	Consum de cada minut del dia següent
Optimitzador	SGD
Mida del lot	32 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1	150
Funció d'activació a la *output layer*	Cap

Taula 5.5. Resum de les característiques de la prova 1.3

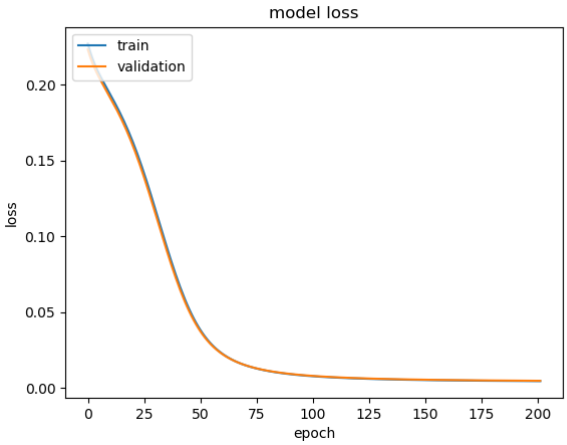

Figura 5.10. Loss de la prova 1.3

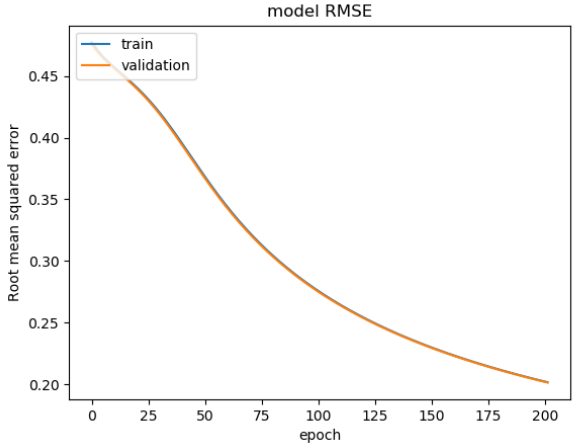

Figura 5.11. RMSE de la figura 1.3

Test *loss*	0.00481
Test RMSE	0.2013
Nombre d'iteracions	199

Taula 5.6. Dades sobre el conjunt de test

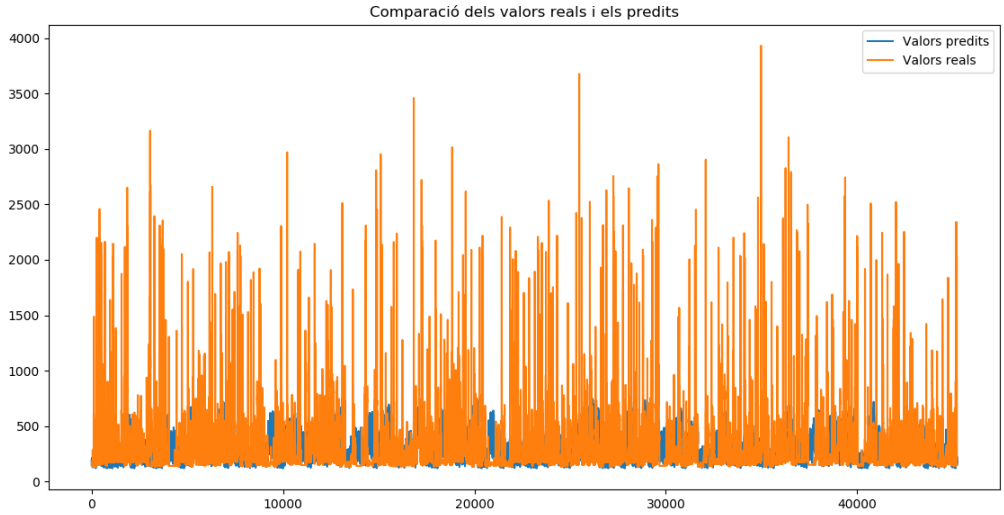

Figura 5.12. Prediccions de la prova 1.3

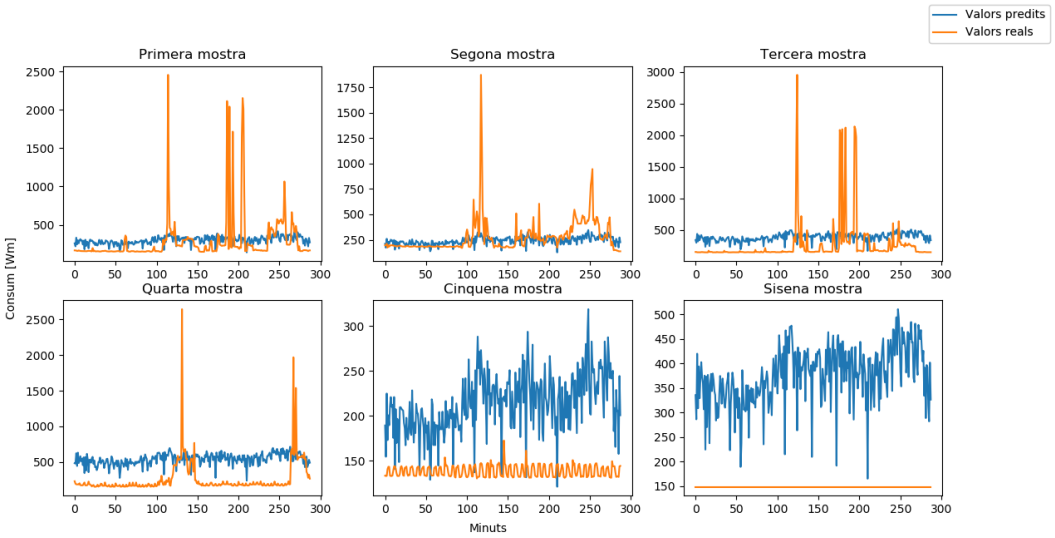

Figura 5.13. 6 mostres de la prova 5.1.3

5.1.4. Prova amd dades reescalades i optimitzador Adam

Contingut matriu *input*	Consum de cada minut, minut, dia anterior
Contingut matriu *target*	Consum de cada minut del dia següent
Optimitzador	Adam
Mida del lot	32 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU

Nombre neurones *hidden layer* 1	150
Funció d'activació a la *output layer*	Sigmoide

Taula 5.7. Resum de les característiques de la prova 5.1.4

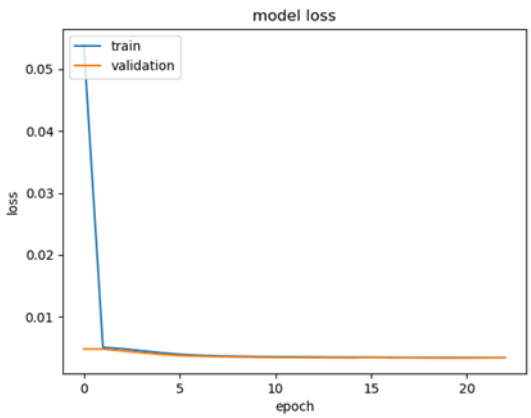

Figura 5.14. Loss de la prova 1.4

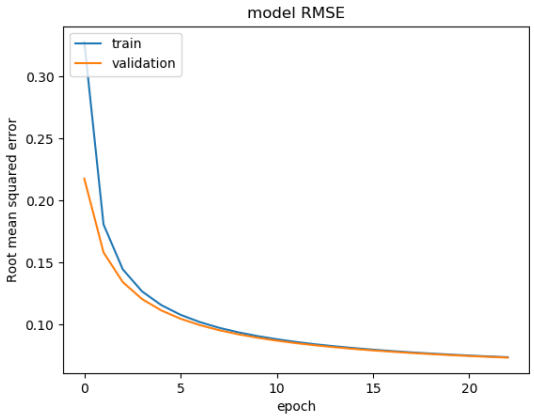

Figura 5.15. RMSE de la prova 1.4

Test *loss*	0.00371
Test RMSE	0.07333
Nombre d'iteracions	20

Taula 5.8. Dades sobre el conjunt de test

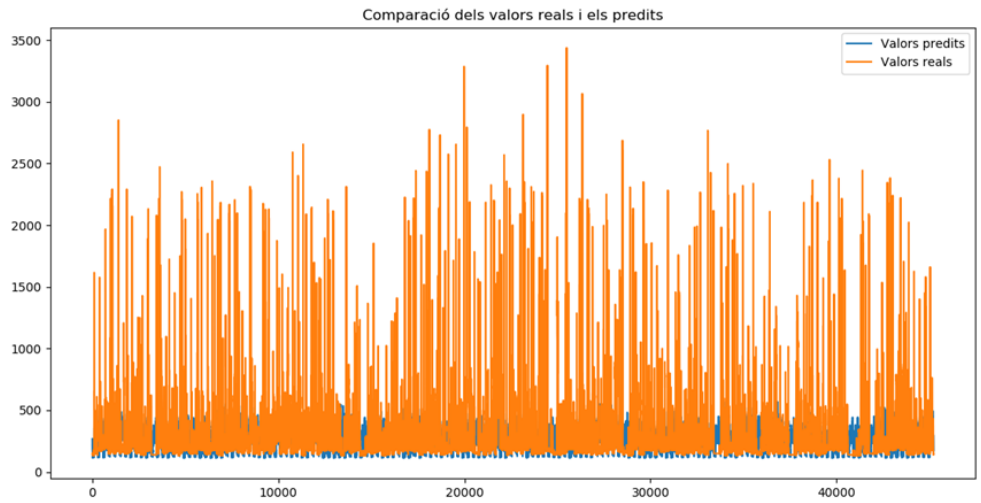

Figura 5.16. Prediccions de la prova 1.4

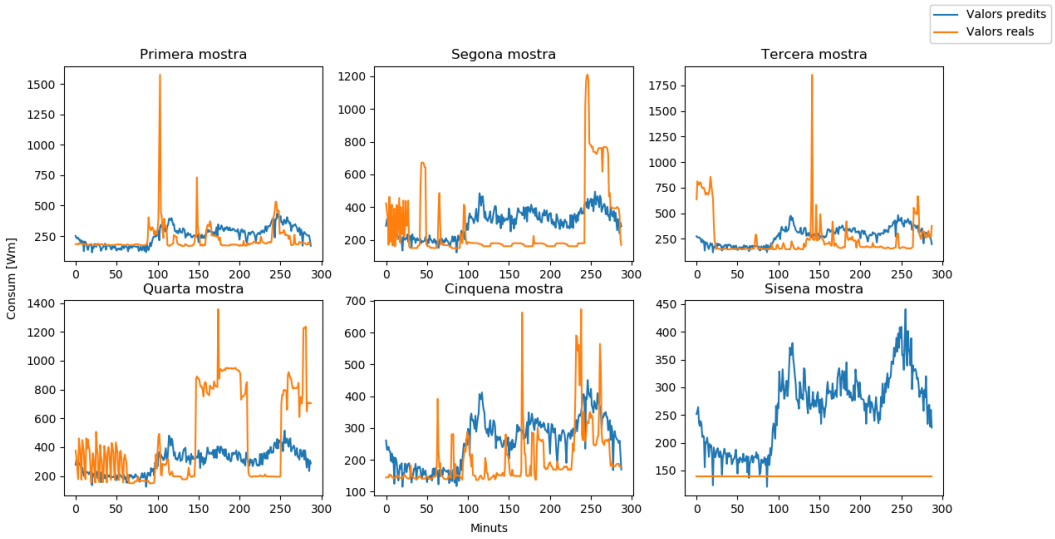

Figura 5.17. 6 mostres de la prova 1.4

5.1.5. Conclusions de la prova 1

Com es pot observar, en totes les subproves realitzades els resultats obtinguts han estat molt semblants i molt poc precisos, bàsicament degut a la forma a la qual s'han introduït les dades: utilitzar les dades d'un dia per predir les del dia següent no sembla en cap cas bona idea, ja que la xarxa tendeix a aprendre una forma el més general possible que s'adapti als hàbits de consum més comuns, però que és incapaç d'adaptar-se a qualsevol canvi i, per tant, proporciona unes estimacions molt poc acurades. Això també és degut al petit nombre de dades de què es disposa si es fa servir aquesta estructura: únicament es tenen, sumant els tres conjunts (entrenament, validació i test), 1572 mostres, el que és un nombre reduït de dades per a una xarxa d'aquestes característiques.

Les conclusions que es poden extreure d'aquesta prova són les següents: primer de tot, s'observa que fer una única predicció diària per estimar el consum elèctric no és una bona solució. A més, també es pot apreciar que l'optimitzador *Adam* fa convergir les solucions molt més ràpid que l'optimitzador *SGD*, fet que pot ser de vital importància al cas de tenir un gran volum de dades. A més, es pot veure que per a poder fer servir les dades normalitzades no s'ha posat cap funció d'activació a la capa de sortida, ja que les dades no estan limitades a cap rang (únicament s'aconsegueix que tinguin una mitjana centrada en zero i una desviació estàndard unitària). Això pot fer que, en proves posteriors, les prediccions no siguin molt precises, ja que el fet de què no s'assigni una funció d'activació a les neurones comporta que aquestes es comportin de manera lineal, mentre que si s'utilitza una funció d'activació adient es pot aconseguir un comportament més complex que ajudi a

predir situacions més complicades de forma correcta. Una part d'aquest problema es pot observar en les prediccions obtingudes amb les dades normalitzades: hi ha valors negatius de consum en aquestes proves, fet que a la realitat és impossible. Tot i que utilitzant funcions d'activació més complexes es podria arribar a solucionar aquest problema, és més pràctic fer servir les dades reescalades i utilitzar les funcions d'activació corresponents a aquest cas, al qual ja es coneix que totes les dades estaran limitades al rang comprès entre 0 i 1, coincideix amb el rang de valors de sortida que té la funció Sigmoide.

Per tant, a la següent prova es canviarà l'estructura de les matrius d'*input* i *target* per una que pugui donar millors resultats. A més, a partir d'ara es faran servir únicament les dades en format reescalat.

5.2. Prova 2: Actualització de la predicció cada cinc minuts

Respecte a la prova anterior, la gran diferència radica en el model de predicció que es realitza: mentre que al cas anterior s'actualitza únicament una vegada al dia, ara s'actualitza cada cinc minuts, el que es tradueix en que les matrius d'*input* i *target* presenten una estructura diferent, que és la que es mostra a la següent imatge:

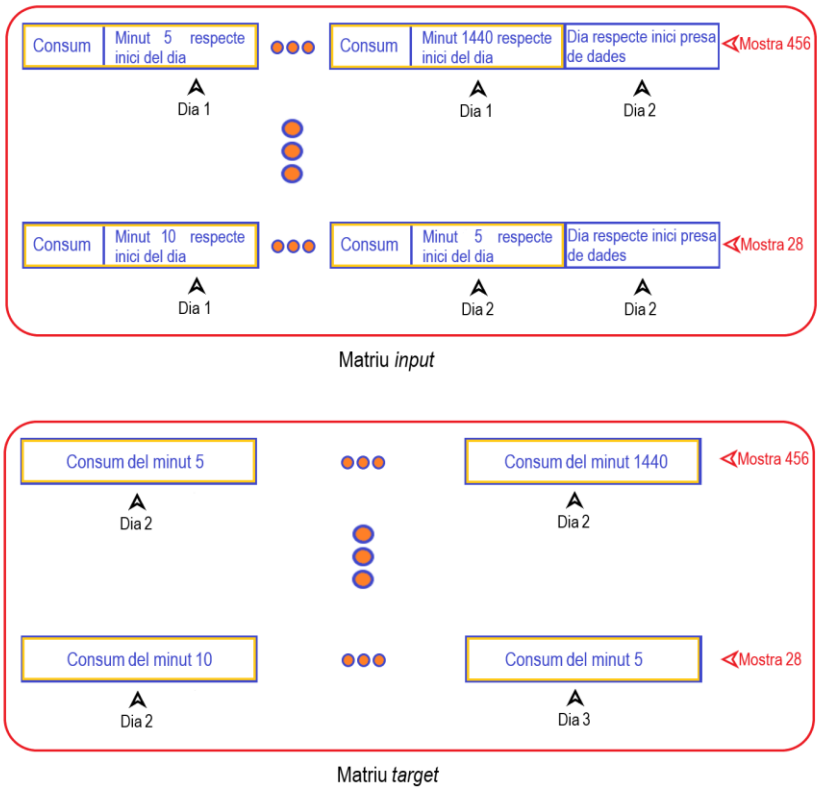

Figura 5.18. Matrius d'input i target

Com es pot observar, les mostres es van desplaçant cada cinc minuts. A més, també es pot observar que s'ha canviat l'estructura de les dades: mentre que a la primera prova es tenia a l'entrada una estructura de tensor (o matriu) 3D, on totes els vectors amb informació de cada minut d'un dia s'agrupaven en una matriu que contenia tots els d'aquell dia i a la vegada totes aquestes estaven contingudes dins d'un altre tensor, ara es disposa d'un vector amb tota la informació de les 24 hores.

Un objectiu és determinar com evolucionen la qualitat de les prediccions en funció del nombre de neurones, ja que no es pot saber si la mida és idònia fins que s'hagin realitzat diferents proves: per tant, s'avaluarà com són les prediccions en funció d'aquest factor. Un altre dels objectius d'aquesta prova és avaluar com és la variació del comportament de la xarxa neuronal en afegir altres valors que poden ser d'interès i que poden ajudar a millorar la qualitat de les prediccions al estar relacionats amb el consum registrat aquell dia: aquests són el dia de la setmana, al mes al qual es produeix i la informació meteorològica. Aquests aspectes s'expliquen amb major nombre de detalls dins de l'apartat corresponent.

Per considerar que l'entrenament ha acabat, si bé a la prova anterior es considerava que l'entrenament havia finalitzat quan durant 3 etapes l'indicador sobre la *loss* del conjunt de validació ha empitjorat, tot restaurant els pesos als quals s'ha obtingut millors prediccions. Per aquest cas es va realitzar una prova prèvia per tal de comprovar si amb aquestes 3 etapes d'espera era suficient per haver assolit el mínim en l'error, es va comprovar que no era així: per tant, d'ara en endavant s'ha utilitzat com a criteri per aturar l'entrenament que la *loss* del conjunt de validació no hagi millorat o directament hagi empitjorat durant 10 etapes seguides. Tal i com es realitzava a la prova anterior, es restauren els pesos de la etapa on es registrava menor error.

En quant a la *batch size,* degut a què ara es disposa d'un número de dades molt més gran, s'ha augmentat de 32 (paràmetre per defecte en la creació de xarxes neuronals al utilitzar Keras) fins a 128 per a agilitzar els càlculs. És un paràmetre del qual no es sap exactament com pot ser la seva influència en la qualitat de les prediccions, ja que hi ha estudis que diuen que això pot ajudar a obtenir uns resultats més acurats[30][31], tot i que aquest procediment pot resultar més lent perquè, dins de cada iteració, es necessiten més passes per tal d'arribar als resultats correctes. No obstant això, hi ha investigadors que afirmen que agafar aquestes petites "porcions" de dades pot provocar gran inestabilitat degut a la gran variància en el descens del gradient estocàstic i que provoca molt soroll en el mateix, el que pot portar a prediccions de pitjor qualitat [32]. Tot i això, no s'ha considerat necessari estudiar aquest

aspecte, ja que les variacions en la precisió dels resultats poden ser petites i de difícil avaluació.

Els gràfics i resultats generats de cada una de les subproves són els mateixos que per a la primera prova, no obstant això, hi ha una variació en la forma de visualitzar-los: com que ara el nombre de dades és més gran, si es fa una visió general dels resultats és difícil veure com són les prediccions, per tant, ara es mostraran únicament els sis dies seleccionats de forma aleatòria, a més d'una ampliació del gràfic general de les prediccions.

5.2.1. Prova 2.1: Tria de l'optimitzador

Com a la prova anterior no s'ha pogut aclarir quin dels dos optimitzadors fa mínim l'error de les prediccions, la primera prova realitzada amb aquesta estructura de dades s'ha fet amb la finalitat de comprovar quin optimitzador porta als millors resultats. Totes les característiques de la prova són les definides anteriorment. Un canvi que s'ha realitzat, respecte a la primera prova és el del nombre de neurones de la capa oculta: amb aquesta estructura, es disposa de 577 paràmetres d'entrada i 288 de sortida, per tant, per aquesta primera prova, el nombre es fixa a 350 neurones, ja que una de les recomanacions a l'hora de dissenyar una xarxa neuronal és que el nombre de neurones d'aquesta capa es trobi entre mig del nombre de les d'entrada i de sortida [33]. Tot i això, aquesta tècnica no és un mètode exacte per a determinar si el paràmetre és correcte. Posteriorment es duran a terme proves per establir-ho. A la taula següent es mostren les característiques més importants de la prova:

Contingut matriu *input*	Consum de cada minut, minut, dia anterior, es van desplaçant cada cinc minuts
Contingut matriu *target*	Consum de cada minut del dia següent, es van desplaçant cada cinc minuts
Optimitzador	SGD-Adam
Mida del lot	128 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1	350
Funció d'activació a la *output layer*	Sigmoide

Taula 5.9. Resum de les característiques de la prova 2.1

Els resultats que es mostren en primer lloc corresponen als de la prova utilitzant l'optimitzador SGD i després els de l'optimitzador Adam.

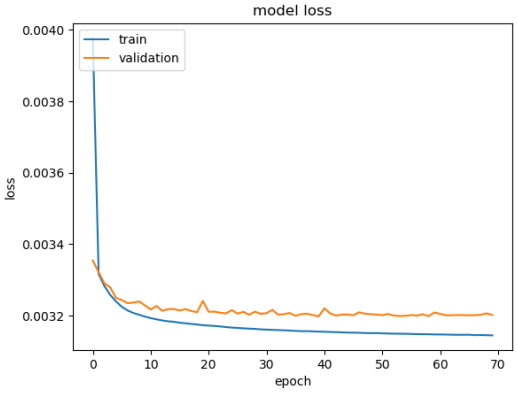

Figura 5.19. Loss de la prova 2.1-SGD

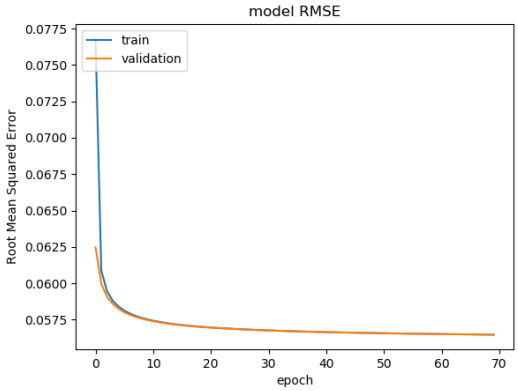

Figura 5.20. RMSE de la prova 2.1-SGD

Test *loss*	0.00370
Test RMSE	0.06404
Nombre d'iteracions	250

Taula 5.10. Dades del conjunt de test de la prova 2.1 -SGD

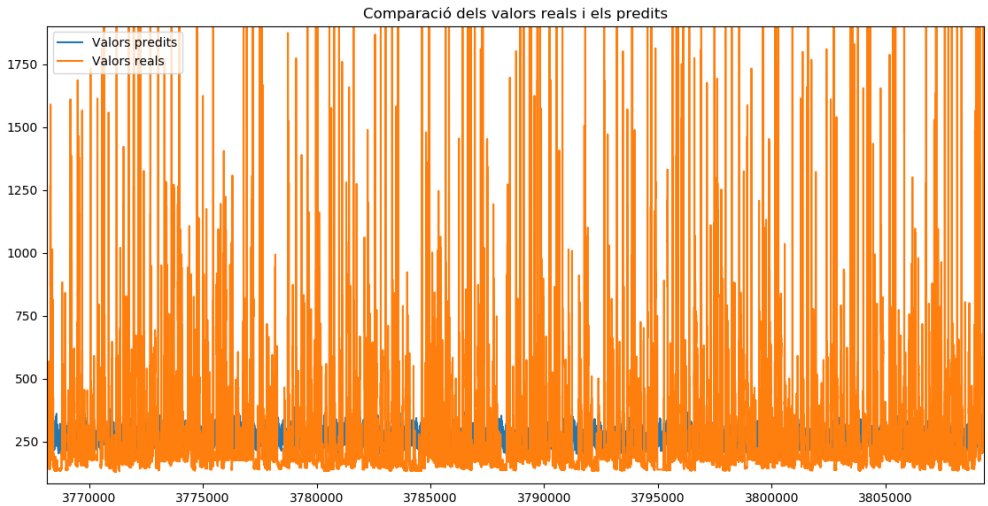

Figura 5.21. Prediccions de la prova 2.1-SGD

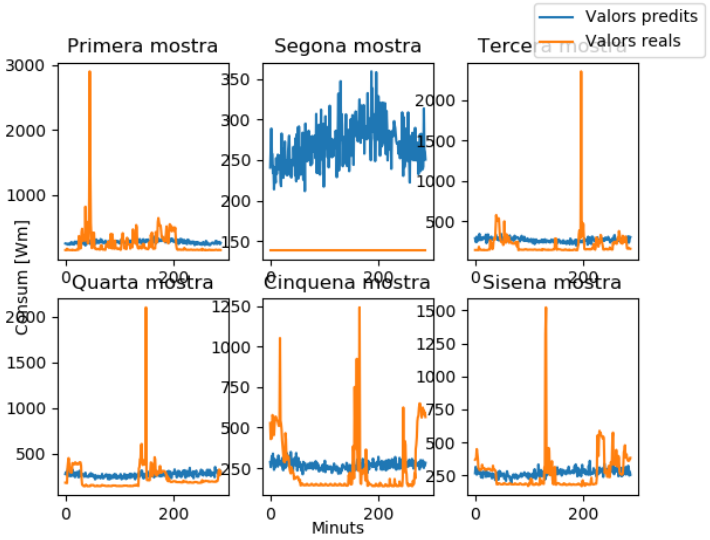

Figura 5.22. 6 mostres de la prova 2.1-SGD

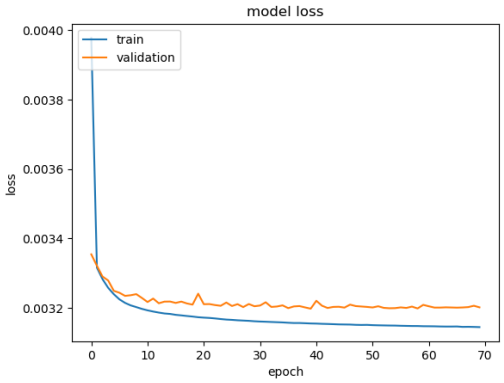

Figura 5.23. Loss de la prova 2.1-Adam

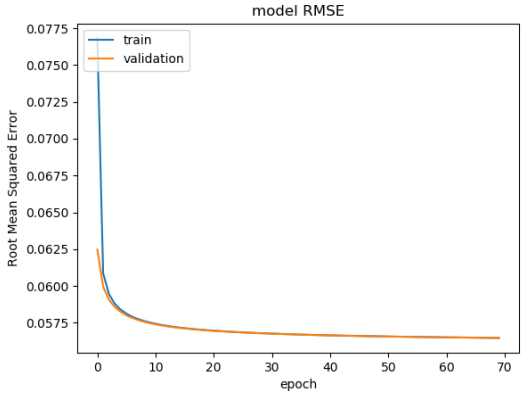

Figura 5.24. RMSE de la prova 2.1-Adam

Test *loss*	0.00320
Test RMSE	0.05646
Nombre d'iteracions	66

Taula 5.11. Dades sobre el conjunt de test de la prova 2.1-Adam

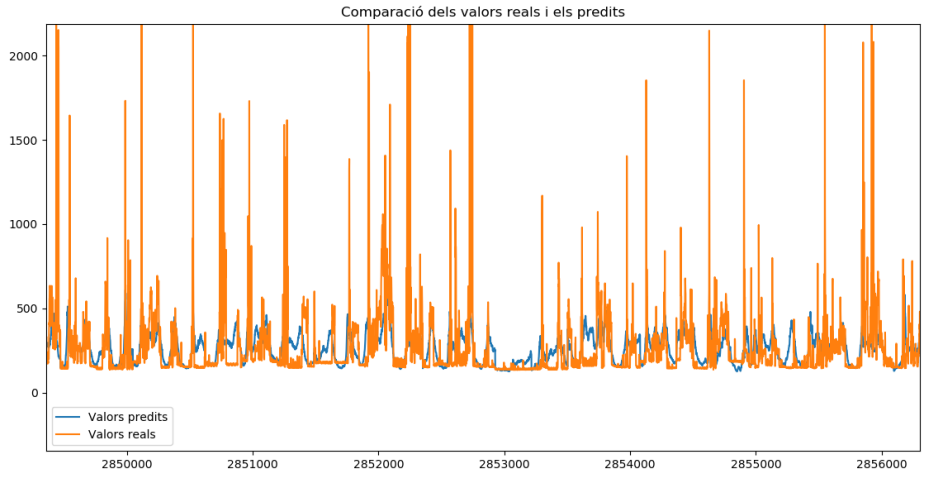

Figura 5.25. Prediccions de la prova 2.1-Adam

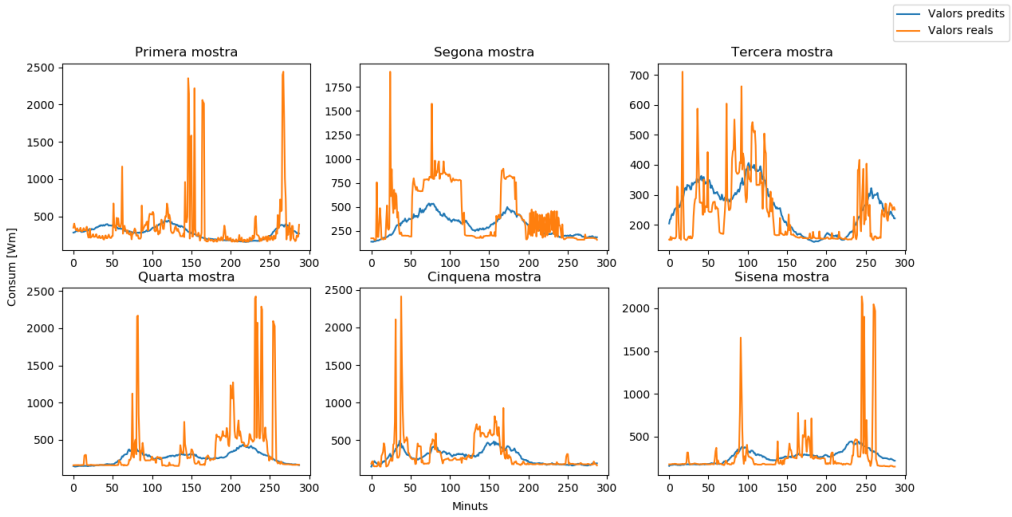

Figura 5.26. 6 mostres de la prova 2.1-Adam

En vista dels resultats obtinguts en aquesta prova, es pot veure un dels problemes que té el primer optimitzador: troba un valor al qual les solucions tenen un mínim local i degut al procés iteratiu de descens estocàstic que segueix, no pot trobar com "sortir" d'ell, provocant que arribi molt aviat a una suposada solució que és errònia. Aquest problema, característic de l'optimitzador SGD, està causat per un problema ja comentat anteriorment, i és la necessitat d'ajustar certs paràmetres del model, concretament la taxa d'aprenentatge, massa gran per aquest model [34] i que porta a una solució incorrecta. Al observar que en l'altre cas es poden obtenir resultats en la bona direcció sense més problemes ni cap modificació addicional, i amb un nombre d'iteracions menor (i, per tant, en menys temps de càlcul), a partir d'ara es farà servir Adam com a optimitzador. Per tant, les proves següents consistiran en modificar altres valors de la xarxa, com el nombre

de neurones a les capes ocultes, el nombre de capes i les dades que s'utilitzen per realitzar les prediccions.

5.2.2. Prova 2.2: Nombre de neurones en una única capa

En aquesta prova, s'avalua quin impacte té la variació del nombre de neurones emprades a la capa oculta de la xarxa neuronal mentre es mantenen la resta de factors iguals. Per a visualitzar quin impacte té aquest canvi, es mostra una gràfica la qual conté quin és el RMSE obtingut per al conjunt de test per a cadascuna de les proves. El nombre d'iteracions al qual s'ha arribat a la solució no es considerarà un factor determinant, ja que pot presentar variacions sense que aquestes estiguin causades pel nombre de neurones, sinó perquè, al estar barrejades les mescles, el fet de què siguin tractades cada vegada en un ordre diferent pot donar lloc a grans diferències. També es visualitzarà, com als casos anteriors, com resulten les prediccions per una selecció aleatòria de 6 mostres. El nombre de neurones s'incrementarà de forma gradual de 50 unitats, partint de 50 neurones i arribant fins a 700. Així es pot avaluar quin és l'impacte que aquest factor té i triar la mida més apropiada per a les proves posteriors. Les subproves contingudes dins de la prova 2.2 s'ordenen de forma creixent: és a dir, la 2.2.1 és la que s'ha realitzat amb 50 neurones a la capa oculta, la 2.2.2 és la que conta amb 100 neurones i així successivament. Aquesta numeració serveix per ubicar les figures corresponents a aquesta prova i que no figuren a la memòria per facilitar la seva lectura, però es troben a l'annex adjunt. A continuació es mostra una taula on s'indiquen els paràmetres bàsics de la xarxa neuronal, comuns a totes les subproves de la prova 2.2:

Contingut matriu *input*	Consum de cada minut, minut, dia anterior, es van desplaçant cada cinc minuts
Contingut matriu *target*	Consum de cada minut del dia següent, es van desplaçant cada cinc minuts
Optimitzador	Adam
Mida del lot	128 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1	50-100-150-200-250-300-350-400-450-500-550-600-650-700
Funció d'activació a la *output layer*	Sigmoide

Taula 5.12. Resum de les característiques de la prova 2.2

Per veure com és la evolució de l'error (RMSE) en funció del nombre de neurones s'ha elaborat un gràfic que es mostra a continuació:

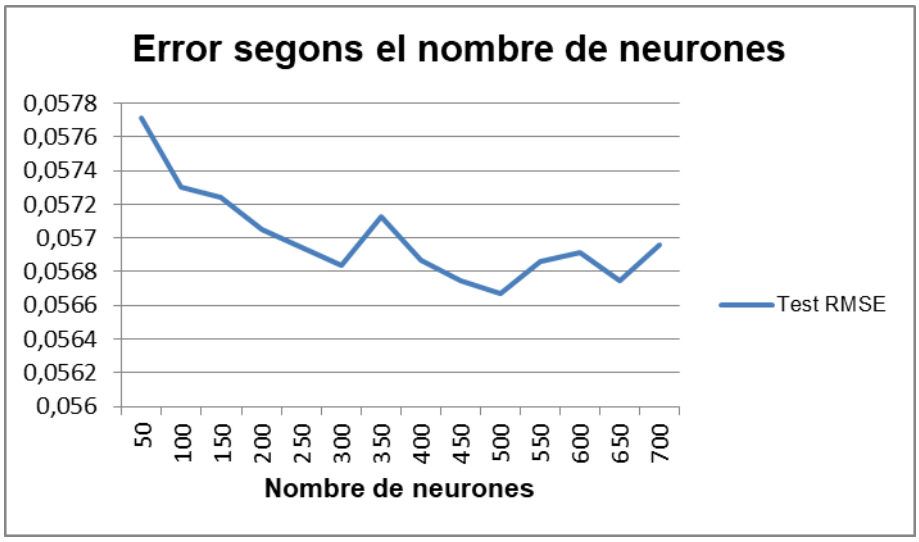

Figura 5.27. Evolució de l'error segons el nombre de neurones

Una vegada s'observen els resultats, tot i que la diferència entre fer servir un nombre de neurones i un altre pot semblar escassa a primera vista, s'ha de tenir en compte que les unitats de l'RMSE són les mateixes a les que es troben les dades reescalades, per tant, una petita variació és significativa. Al principi, l'addició de noves neurones es veu clarament que port a una reducció de l'error i, per tant, a unes prediccions més exactes, i aquesta tendència es manté (tot i registrar un augment puntual de l'error a les 350 neurones) fins a les 500 neurones.

Aquest increment de l'error a les 350 neurones s'ha d'interpretar de forma relativa, ja que es pot observar que, en aquest punt, la *loss* del model es troba en una clara tendència descendent, aquest pic inusualment elevat pot estar produït per les condicions inicials amb les quals s'ha inicialitzat la xarxa, així com quina ha estat la divisió dels valors en els tres conjunts necessaris (cal recordar que això es realitza de forma aleatòria cada vegada que es fa una nova prova de la xarxa i , per tant, no hi ha forma de controlar aquest paràmetre). Es podria donar el cas de què, tornant a executar el mateix programa els valors fossin diferents i l'error fos més baix i concordant amb la tendència seguida per la resta de punts.

A partir de les 500 neurones, es pot observar que l'RMSE torna a registrar, malgrat una davallada en l'error a les 650 neurones, un augment progressiu fins les 700 neurones. Això pot estar causat per problemes d'*overfitting*: al disposar de tantes neurones, hi ha un nombre

excessiu de connexions, i aquestes es destinen a aprendre únicament informació basada en les dades del conjunt d'entrenament i no una generalització que ajudi a predir de forma adequada [35]. Per tant, si el model s'hagués de realitzar amb una única capa oculta, el nombre de neurones a utilitzar a aquesta seria de 500. A continuació es mostren els gràfics de les 6 mostres aleatòries per a la prova amb l'error més baix:

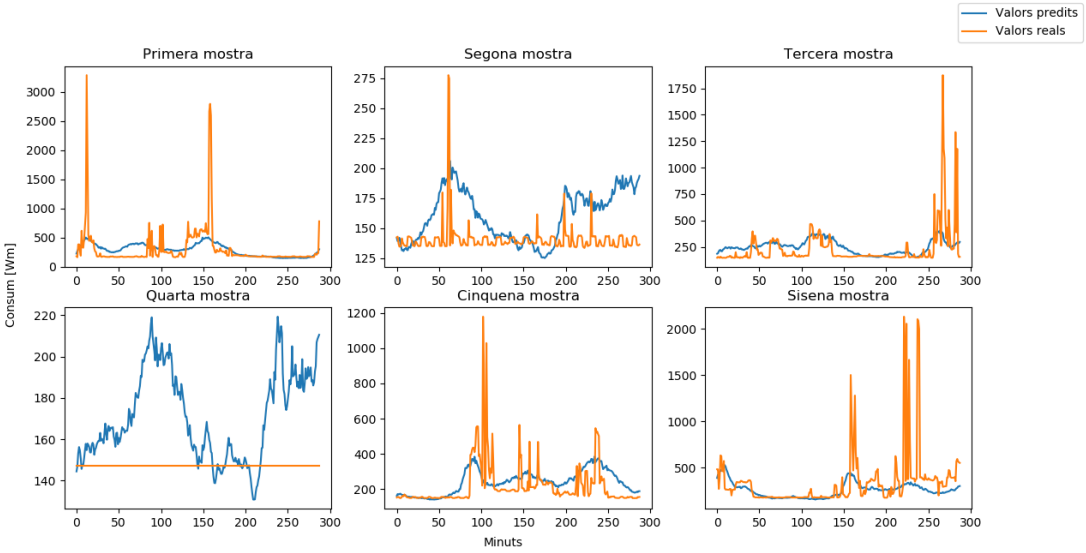

Figura 5.28. 6 mostres de la prova 2.2.10

En aquest gràfic es pot observar com les prediccions comencen a ser fidels a la realitat, i la xarxa es mostra capaç, a la majoria de casos, de trobar la tendència que segueix el consum al llarg del dia. A més, als dies on es veuen diferències més notòries, a més d'observar-se que són dies amb un perfil de consum clarament atípic, es veu que l'escala del gràfic, al modificar-se segons les dades, fa que sembli una desviació més gran de la real.

5.2.3. Prova 2.3: Prova amb una capa oculta addicional

Partint dels resultats de la prova anterior, on s'ha determinat el nombre de neurones que proporcionen el millor resultat possible, s'afegeix una altra capa oculta al model per tal d'observar quina implicació a les prediccions té aquesta modificació. Com s'ha comentat anteriorment, un increment en el nombre de capes pot millorar els resultats si es tracta de conjunts de dades complexos. A vegades, però, el augment del nombre de capes pot portar un augment de complexitat innecessari que realment provoqui pitjors prediccions, al existir el risc de què es produeixi *overfitting* i, per tant, la xarxa aprengui en excés com és conjunt d'entrenament i llavors sigui incapaç de predir sobre unes altres dades correctament.

En aquesta prova s'estudiarà com evoluciona la qualitat de les prediccions amb dues capes ocultes iguals, és a dir, amb la mateixa funció d'activació i amb el mateix nombre de neurones. Com a la prova 2.2, les subproves s'han enumerat seguint un ordre creixent: 2.2.1 per a la xarxa neuronal amb 50 neurones a cada capa, 2.2.2 si el nombre de neurones a cada capa és de 100 i així successivament. A continuació es mostra una taula amb el resum de les característiques de la prova, i posteriorment, tal i com s'ha realitzat a la prova anterior, es mostra una taula amb els valors de RMSE obtingut pel conjunt de test en funció del nombre de capes ocultes emprades.

Contingut matriu *input*	Consum de cada minut, minut, dia anterior, es van desplaçant cada cinc minuts
Contingut matriu *target*	Consum de cada minut del dia següent, es van desplaçant cada cinc minuts
Optimitzador	Adam
Mida del lot	128 mostres
Nombre *hidden layers*	Una *hidden layer*
Funcions d'activació a les *hidden layers*	ReLU
Nombre neurones *hidden layer* 1-2	50-100-150-200-250-300-350
Funció d'activació a la *output layer*	Sigmoide

Taula 5.13. Resum de les característiques de la prova 2.3

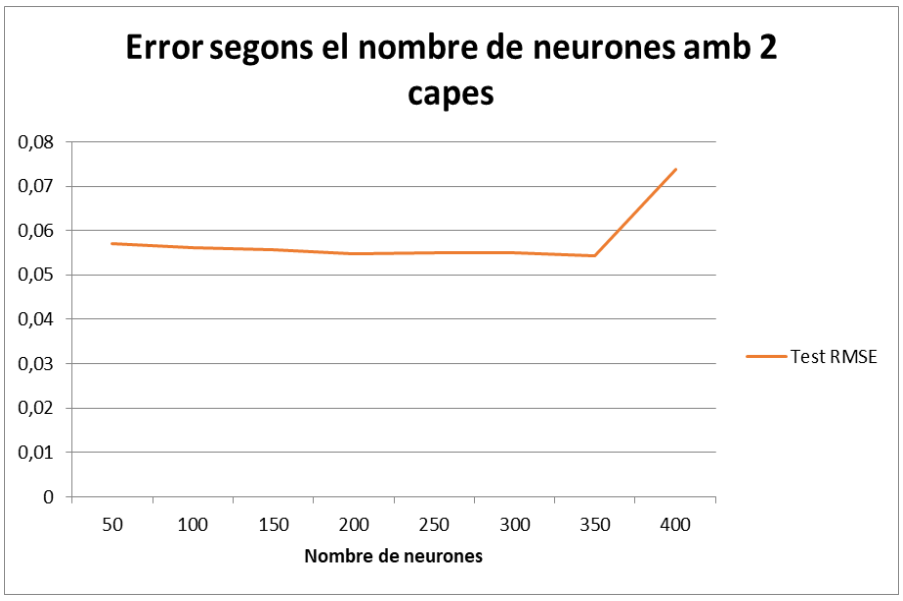

Figura 5.29. Evolució de l'error segons el nombre de capes

Com es pot observar a la figura superior, si bé el fet d'haver afegit una capa oculta addicional suposa una disminució de l'error, això pot comportar problemes si el nombre de neurones a cada una de les capes és massa gran. A les 300 neurones es presenta un lleuger increment en l'error (que pot estar provocat, tal i com succeïa a la prova 2.2, per les condicions d'inicialització de la xarxa), aquest baixa a les 350 neurones i es torna a provocar un increment a les 400 neurones. Un aspecte important d'aquesta prova és observar el què succeeix a aquest valor i que es visualitza amb més claredat a l'extracte de 6 mostres aleatòries:

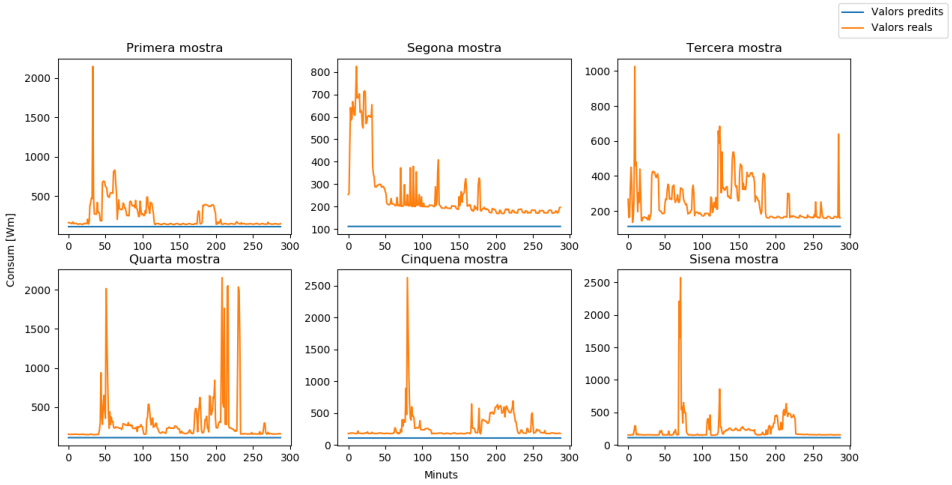

Figura 5.30. 6 mostres de la prova 2.3.8

En aquest cas, la xarxa neuronal es mostra incapaç d'aprendre. Un augment indiscriminat del nombre de neurones pot treure a la llum un dels principals inconvenients que tenen les xarxes amb més d'una capa oculta i que utilitzen en aquestes la funció d'activació ReLU. Aquest problema es coneix com a *dying ReLU*: això significa que totes les neurones que tenen aquesta funció d'activació es tornen inactives (és a dir, presenten com a valor de sortida 0). Com és un problema que succeeix quan moltes neurones no s'activen mai, és més probable que es produeixi quan es tenen massa neurones a cada capa. Per solucionar aquest problema es proposen diferents solucions: la més simple passa per reduir el nombre de neurones per capa. Hi ha d'altres, però, que impliquen la modificació d'altres paràmetres del model, com bé pot ser la reducció de la taxa d'aprenentatge [36] o bé realitzar altres mètodes d'inicialització de la xarxa neuronal [37]. Per tant, es decideix no realitzar més proves consistents en augmentar el nombre de neurones per capa. A les següents proves es tria aquesta l'estructura amb dues capes i 350 neurones per capa. A continuació es mostra la figura amb una mostra de les prediccions per a 6 dies diferents:

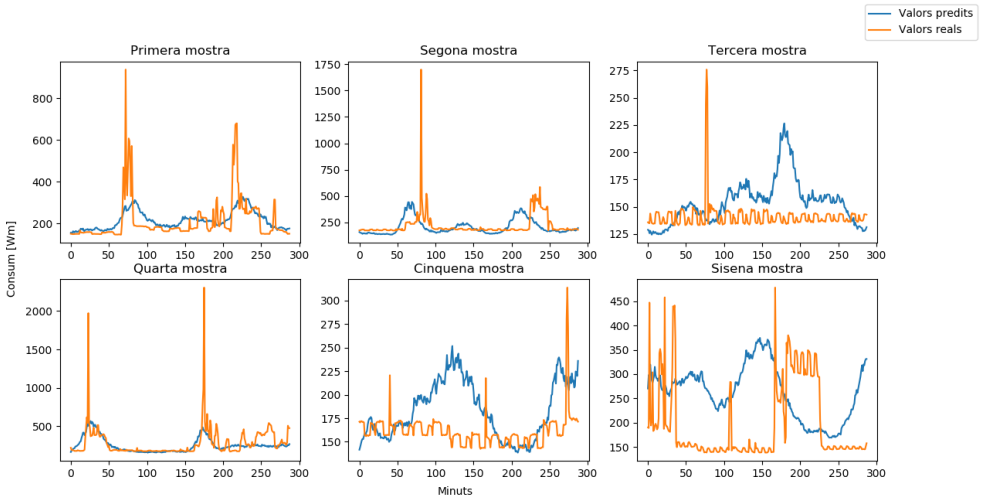

Figura 5.31. 6 mostres de la prova 2.3.7

Es pot observar que hi ha dies als quals la xarxa no és capaç de predir com és el perfil de consum, donant lloc a resultats poc acurats. Això és causat per diferents motius: primer de tot, l'escala del gràfic fa que les diferències entre els valors reals i els predits es vegin amplificades: segon, es tracta de dies amb un perfil de consum anòmal que pot estar causat per diferents motius, com que es tracti d'un període al qual no hi hagi cap persona a l'habitatge i l'únic consum que es realitzi estigui causat per elements que es mantinguin endollats al corrent elèctric. En aquest darrer aspecte té molt a veure les dades introduïdes a la xarxa neuronal, que només té com a indicador temporal el minut del dia i el dia respecte al qual es van començar a prendre les dades, el que fa que no tingui cap altre indicador sobre els dies als quals pot haver una demanda elèctrica que surti del perfil comú. Per tant, a les següents etapes s'introduiran més dades de caire temporal per avaluar el seu impacte als resultats.

No obstant això, en realitat els resultats per a aquesta configuració de la xarxa neuronal no són inexactes, com pot semblar a partir del què s'observa a la Figura 5.30. Com s'ha especificat anteriorment, els dies seleccionats als gràfics que contenen 6 mostres s'han triat de forma aleatòria, el que pot fer que a vegades es seleccionin per visualitzar prediccions en dies amb perfils de consum anòmals. Si observem la Figura 5.31, que conté un detall del gràfic amb els resultats amb 350 neurones a les dues capes ocultes, aquests són molt més propers a la realitat:

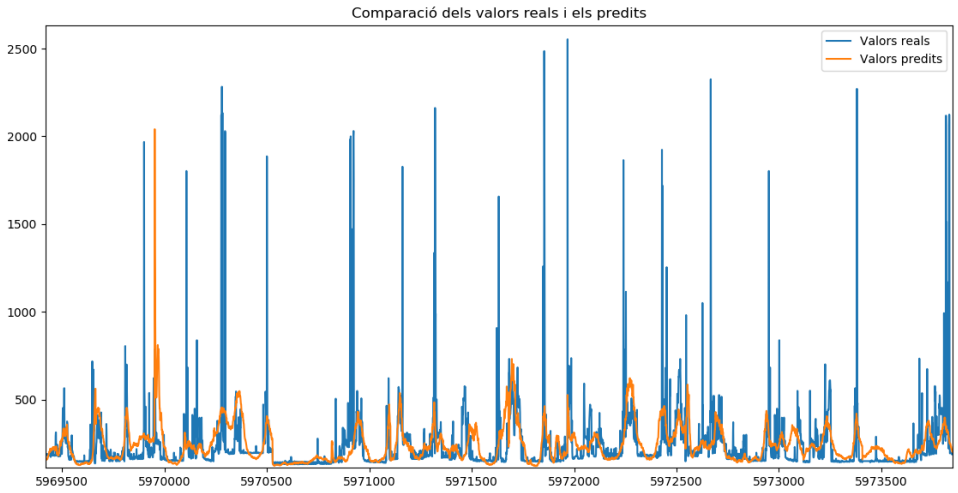

Figura 5.32. Prediccions de la prova 2.3.7

5.2.4. Prova 2.4: Prova incloent el dia de la setmana

A aquesta prova s'inclou el dia de la setmana del qual es vol predir el consum, una dada que ja estava inclosa a les dades de consum inicials. En teoria, la inclusió d'una dada com aquesta hauria de millorar els resultats, ja que és una dada important que pot indicar períodes amb un perfil de consum diferent: usualment la manera a la qual s'utilitza l'electricitat a un habitatge és diferent, per exemple, durant el cap de setmana que a la resta de la setmana. Per tant, el nombre d'entrades a la xarxa neuronal ha augmentat en un, passant de 577 a 578. La resta de paràmetres de la xarxa i l'estructura de la xarxa es mantenen iguals que a la prova 2.3. A continuació es mostra una taula on figuren els valors de l'RMSE per a la prova 2.3.7 i la 2.4:

Sense el dia de la setmana	0.05424
Amb el dia de la setmana	0.05356

Taula 5.14. RMSE per a les proves 2.3.7 i 2.4

Com era d'esperar, l'addició d'aquesta dada ha disminuït l'error, per tant, ha suposat una millora en la qualitat de les prediccions. Al cas de les dades sense el dia de la setmana, el nombre d'iteracions realitzades fins que ha convergit ha estat de 113, i per l'altre ha estat molt semblant, de 112. Per tant, afegir el dia de la setmana no implica cap diferència al temps de convergència de l'algorisme. En vista d'aquests resultats, és clar que aquesta

dada és significativa i es manté a les següents estudis. A continuació es mostren els gràfics generats a aquesta prova:

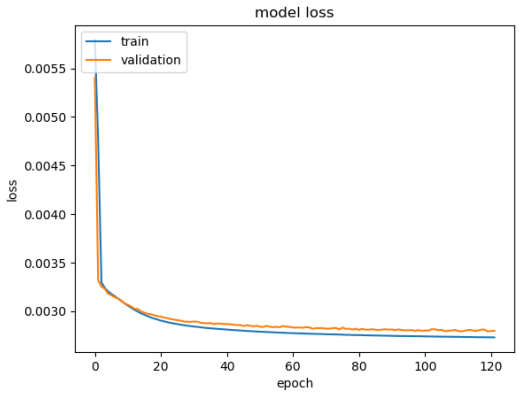

Figura 5.33. Loss de la prova 2.4

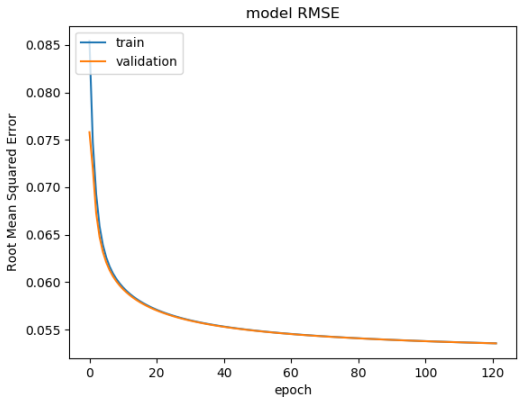

Figura 5.34. RMSE de la prova 2.4

Test *loss*	0.00279
Test RMSE	0.05356
Nombre d'iteracions	112

Taula 5.15. Dades sobre el conjunt de test de la prova 2.4

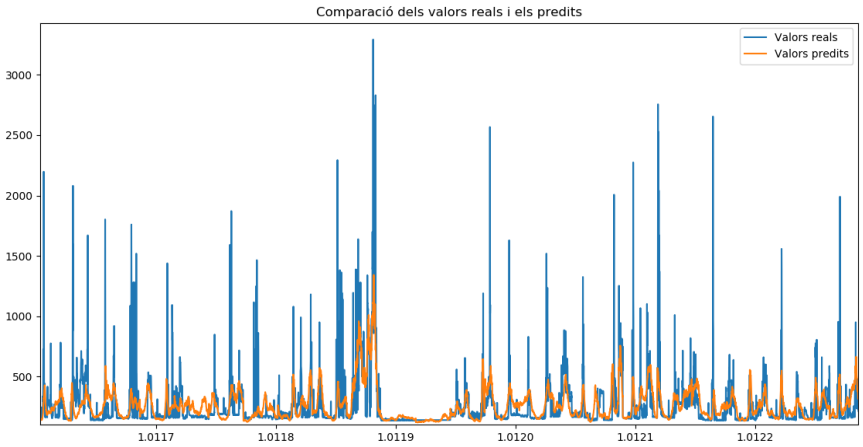

Figura 5.35. Prediccions de la prova 2.4

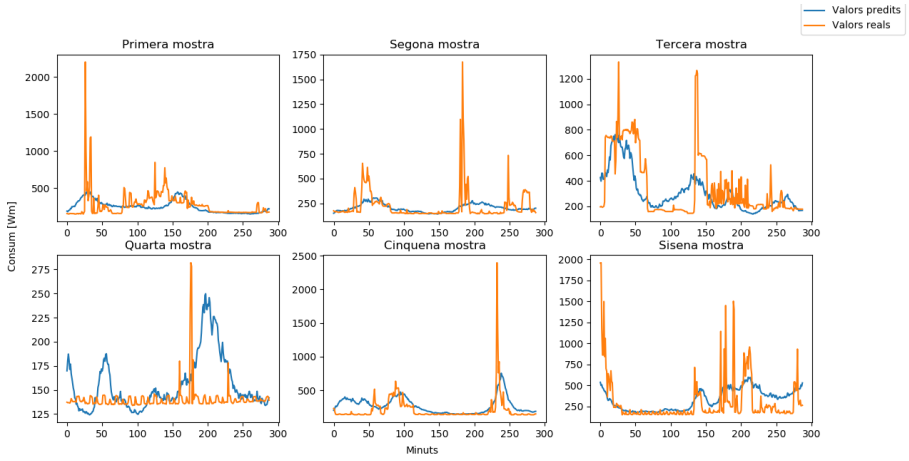

Figura 5.36. 6 mostres de la prova 2.4

5.2.5. Prova 2.5: Prova incloent el dia de la setmana i el mes

Aquesta prova segueix la mateixa lògica que l'anterior, s'afegeix un indicador temporal (en aquest cas, el mes al qual es registra el consum) per tal d'aportar informació extra que millori els resultats. En aquest cas, el mes hauria d'aportar informació molt valuosa sobre certs aspectes com, per exemple, els mesos als quals hi ha vacances (quan el consum hauria de ser més baix) o poder diferenciar entre els mesos d'hivern i d'estiu. Com que es disposa dels hàbits de consum de les persones que viuen al pis, es sap que la calefacció funciona amb gas però tenen aparells d'aire condicionat: per tant, el consum a l'estiu és més gran. Aquesta informació la pot aportar el mes. Com a punt de partida d'aquesta prova s'agafa l'anterior, on s'incloïa el dia de la setmana. Tots els paràmetres es mantenen iguals que per al cas anterior, la única diferència és que ara es tenen 579 dades d'entrada a la xarxa neuronal en comptes de 578. La taula següent compara l'RMSE per a les proves 2.4 i 2.5:

Sense el mes	0.05356
Amb el mes	0.05323

Taula 5.16. RMSE per a les proves 2.4 i 2.5

Es pot veure que l'error disminueix però en molt menor mesura que si es compara les proves que contenia les dades sense i amb el dia de la setmana. Ara, el nombre d'iteracions realitzades és de 120 enfront de les 112 de la prova anterior. Això no és una gran diferència, i pot estar en part causada per l'aleatorització inicial de les dades o per les condicions d'inicialització de la xarxa. Tenint en compte aquests factors, com que l'addició d'aquesta dada, malgrat no suposar una gran diferència en la qualitat de les prediccions, com que tampoc representa un gran increment en la complexitat del model, es decideix incloure per a futures proves. A continuació es mostren les figures generades a la prova 2.5:

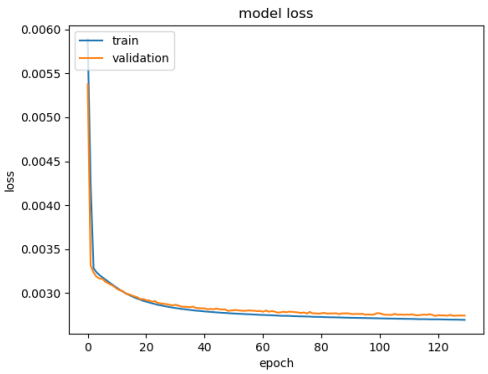

Figura 5.37. Loss de la prova 2.5

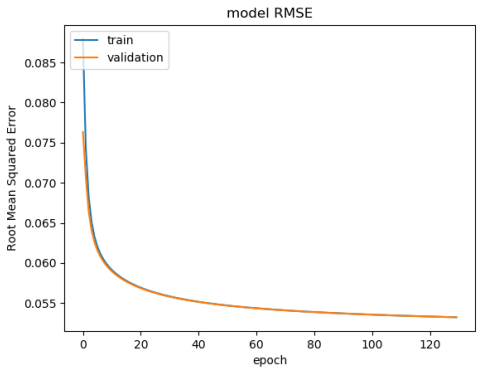

Figura 5.38. RMSE de la prova 2.5

Test *loss*	0.00274
Test RMSE	0.05323
Nombre d'iteracions	120

Taula 5.17. Dades sobre el conjunt de test de la prova 2.5

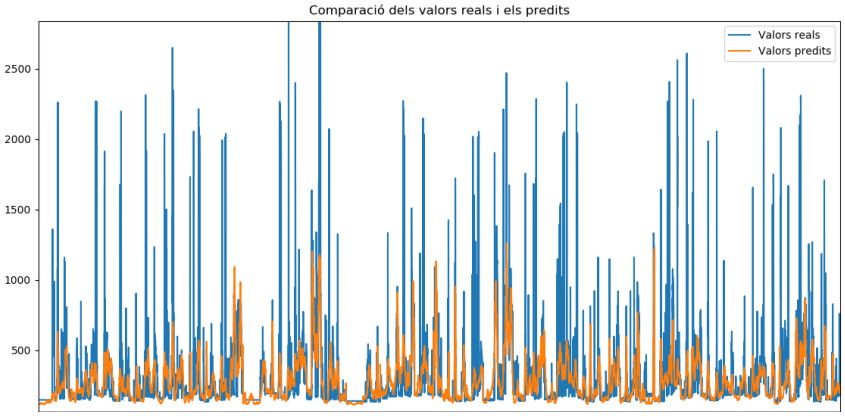

Figura 5.39. Prediccions de la prova 2.5

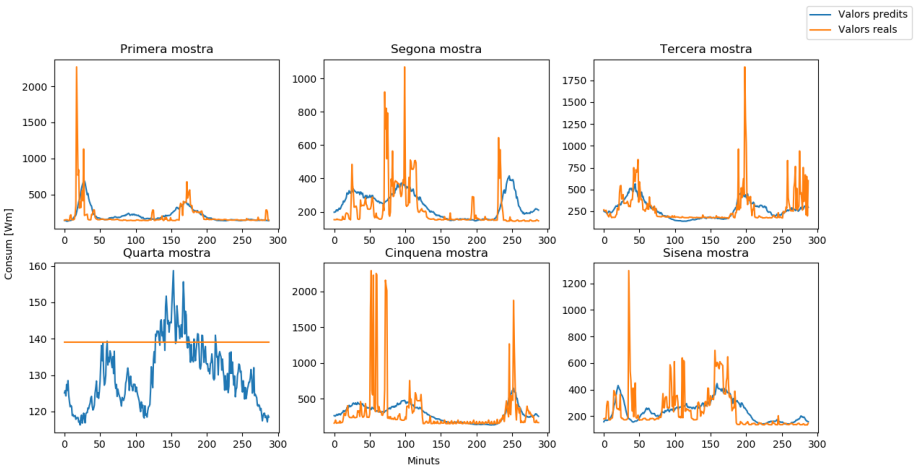

Figura 5.40. 6 mostres de la prova 2.5

5.2.6. Prova 2.6: Prova incloent les dades meteorològiques

Per últim, la darrera prova que es fa és la d'incorporar les dades meteorològiques sobre el dia del qual es vol predir el consum. Aquestes dades, com s'ha explicat anteriorment, es van descarregar de la pàgina Web de l'AEMET i s'han incorporat a les que ja es tenien sobre el consum elèctric de l'habitatge. Concretament, s'han incorporat tres dades de cada dia, totes elles referents a la temperatura: la temperatura màxima, la temperatura mitjana i la temperatura mínima del dia. Amb això s'espera poder millorar encara més els resultats

obtinguts. La tasca que s'ha realitzat en ambdós casos és la d'eliminar la informació meteorològica dels dies als quals no hi ha dades de consum. No obstant això, durant l'anàlisi d'aquestes dades meteorològiques es va observar un problema, la presència de dies als quals, tal i com va succeir amb les dades de consum elèctric, no hi ha cap valor registrat. Davant aquest problema, es van plantejar dues maneres d'enfrontar-ho:

- La primera d'elles és la de mantenir els dies que no tenen cap consum elèctric registrat (per tant, com a valors de temperatura presenten un zero per cadascun d'ells) i afegir un *flag* que indiqui si la mostra de consum és vàlida o no.

- La segona és seguir el mateix procediment realitzat per a les dades de consum i esborrar directament els dies als quals no hi ha registre de temperatura (és a dir, eliminar també les mostres de consum corresponents a aquests dies).

Com que *a priori* es desconeix quina de les dues tècniques proporcionarà millors resultats, s'ha decidit calcular-ho de les dues formes i posteriorment avaluar quina és més òptima. A continuació es mostren els gràfics generats per cadascuna de les proves:

5.2.6.1. Prova mantenint els dies sense dades i incloent el *flag*

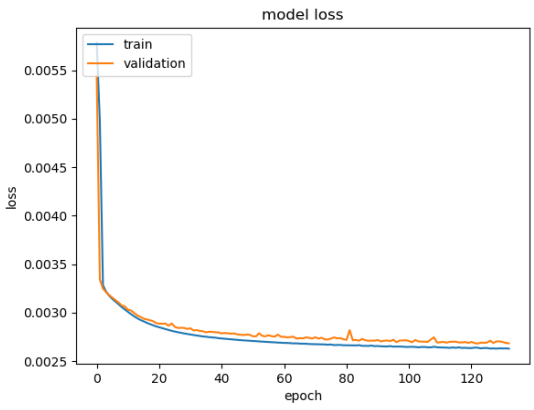

Figura 5.41. Loss de la prova 2.6.1

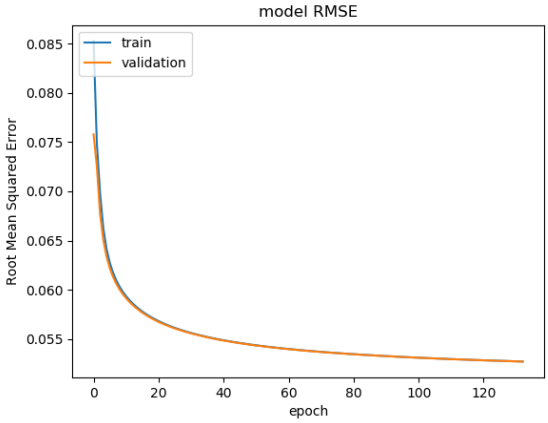

Figura 5.42. RMSE de la prova 2.6.1

Test *loss*	0.00268
Test RMSE	0.05272
Nombre d'iteracions	123

Taula 5.18. Dades sobre el conjunt de test de la prova 2.6.1

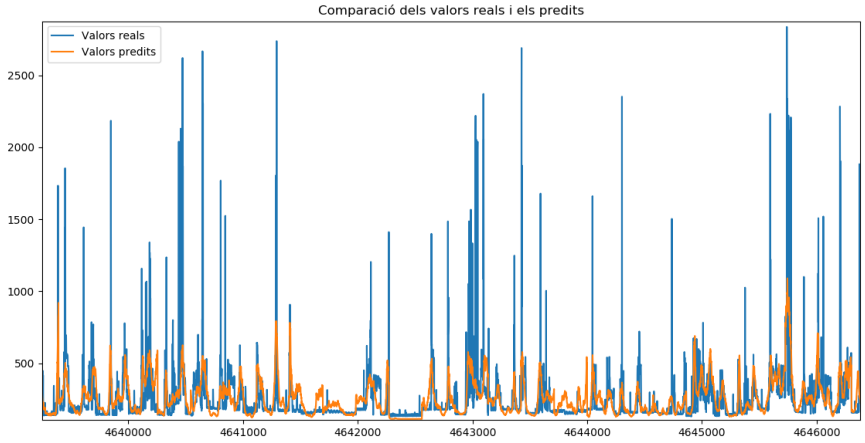

Figura 5.43. Prediccions de la prova 2.6.1

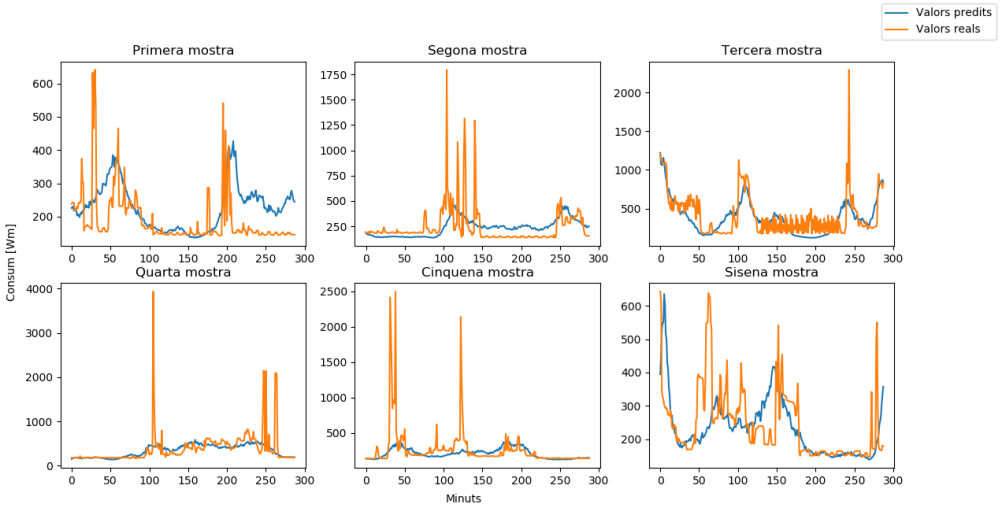

Figura 5.44. 6 mostres de la prova 2.6.1

5.2.6.2. Prova eliminant els dies sense temperatura registrada

En aquest cas, al iniciar l'entrenament, es va registrar el mateix problema que havia succeït a la prova 2.3.8, on gran part de les neurones es trobaven inactives i no s'activaven amb el pas de les iteracions, produint-se a les capes ocultes el que es coneix com a *dying ReLU*, el que fa que la sortida sigui únicament un valor constant. Com s'ha comentat anteriorment, una de les solucions a aquest problema és la de reduir la taxa d'aprenentatge, per tant, aquesta és la via per solucionar el problema. S'ha triat una taxa de 0.0001, suficient per evitar que no convergeixi l'algorisme. No obstant això, aplicant aquesta mesura es perd un dels avantatges de l'optimitzador *Adam,* la taxa d'aprenentatge variable, que fa que el procés iteratiu sigui més ràpid, principalment durant les primeres etapes, on es tria

automàticament una taxa més gran per arribar més ràpid al mínim i posteriorment es redueix per trobar l'òptim amb major precisió. Això farà que les dues proves no siguin totalment comparables, ja que el nombre d'iteracions de la prova 2.6.2 sigui molt més elevat que el de la prova 2.6.1. A continuació es mostren els gràfics generats de la següent prova:

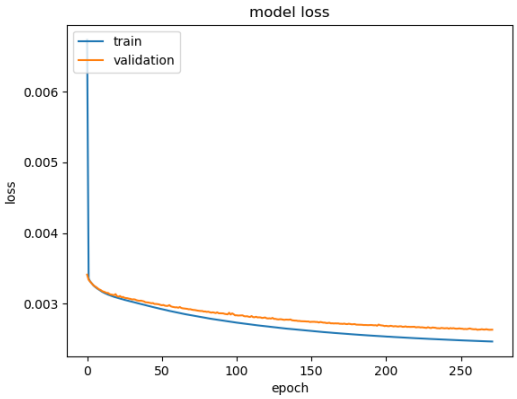

Figura 5.45. Loss de la prova 2.6.2

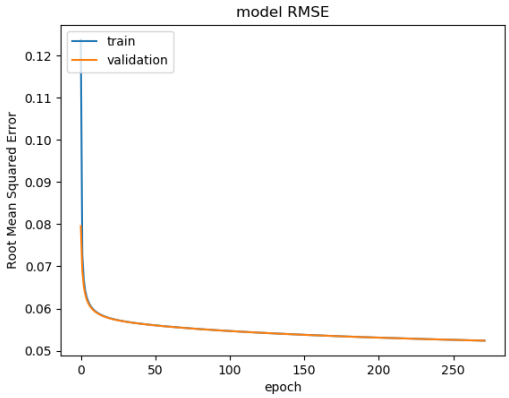

Figura 5.46. RMSE de la prova 2.6.2

Test *loss*	0.00263
Test RMSE	0.05238
Nombre d'iteracions	262

Taula 5.19. Dades sobre el conjunt de test de la prova 2.6.2

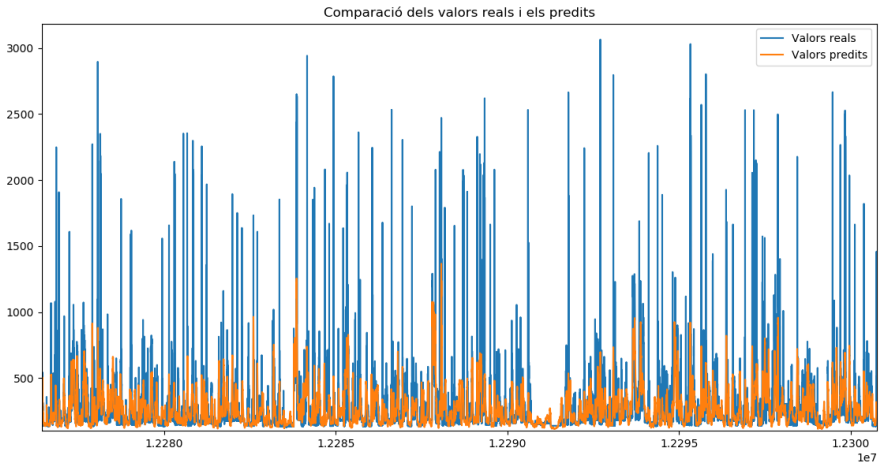

Figura 5.47. Prediccions de la prova 2.6.2

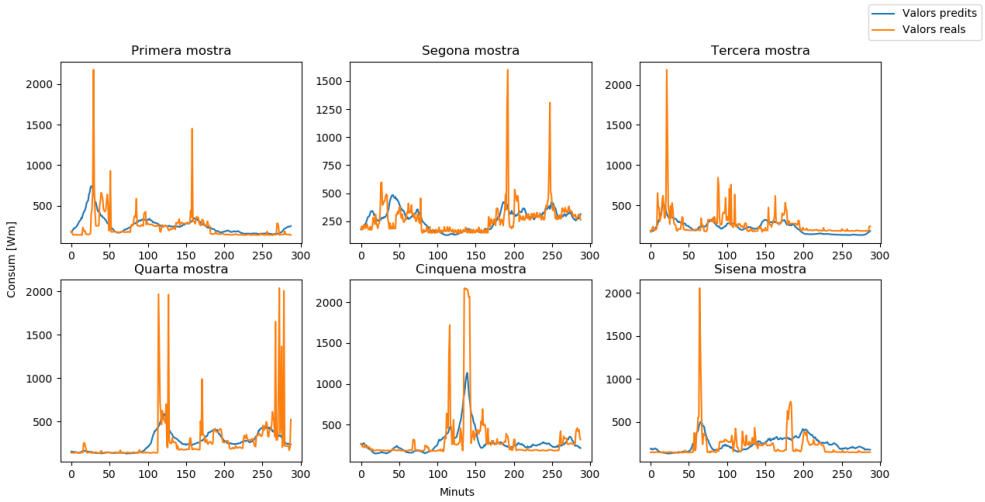

Figura 5.48. 6 mostres de la prova 2.6.2

Si observem les dues prediccions, es pot veure que els resultats són més precisos al segon cas, amb un menor error, suposant pel segon cas una millora respecte al cas sense les dades meteorològiques equivalent a la que s'ha obtingut quan es va passar de 50 a 500 neurones. L'únic inconvenient que presenta és el nombre d'iteracions que es realitzen fins que s'obtenen els resultats, més del doble del que era habitual a les proves precedents (de, aproximadament, 120 iteracions fins a 260) que probablement faci que no compensi per la millora que comporta.

5.2.7. Conclusions de la prova 2

S'ha pogut comprovar que hi ha un gran nombre de paràmetres que influeixen en la qualitat de les prediccions. Hi ha factors com, per exemple, el nombre de neurones quan s'utilitza una única capa, la inclusió del dia de la setmana o la incorporació de les dades de temperatura que sí que marquen la diferència i permeten anar fent les prediccions cada vegada més precisa. Hi ha d'altres, però que no tenen un pes tan clar en els resultats, com el mes o el fet de fer servir dues capes de neurones.

No obstant això, a excepció de la forma d'incloure les dades meteorològiques esmentada a la prova 2.6.2, s'han incorporat al model tots els altres factors i modificacions de paràmetres de la xarxa que comportin una disminució de l'error per petit que sigui aquest, ja que no han provocat cap augment dràstic ni en el temps d'entrenament ni en el nombre d'iteracions.

6. Impacte ambiental

Aquest Treball Fi de Grau té un impacte ambiental baix, un aspecte en consonància amb quin és un dels seus objectius. Durant la realització del mateix, l'únic material utilitzat ha estat l'equip informàtic necessari per a l'elaboració de les diferents proves amb la xarxa neuronal i per a la redacció d'aquesta memòria. Per exemple, no s'ha fet servir en cap moment suport en paper. Els únics aspectes contaminants que poden estar relacionats amb la seva elaboració són dos: el primer pot venir a l'hora d'haver de substituir els ordinadors utilitzats, concretament amb el reciclatge de certs components que, sense el tractament adequat, poden ser nocius pel medi ambient i per l'ésser humà (per exemple, les bateries). L'altre aspecte és el de la contaminació provocada per la generació de l'electricitat necessària per fer funcionar els equips informàtics.

Aquesta contaminació es pot estimar en funció del consum elèctric provocat pel funcionament dels ordinadors. Per a calcular això, s'han tingut en compte les dades subministrades per l'empresa Red Eléctrica de España [38], que proporciona les dades mensuals d'emissió de CO_s causades per la generació d'electricitat fent servir fonts d'energia no renovables. S'han fet servir dos ordenadors, un amb un consum estimat d'uns 90 W i un altre amb un consum estimat de 45 W. El primer s'ha fet servir durant el 80% del temps i l'altre durant el 20%. La despesa total és de 29,160 kW. Al següent gràfic es pot visualitzar com han estat les emissions des de setembre de 2019 fins al desembre de 2019, tot i que les dades pel darrer mes, a data de redacció d'aquesta memòria són incompletes i provisionals:

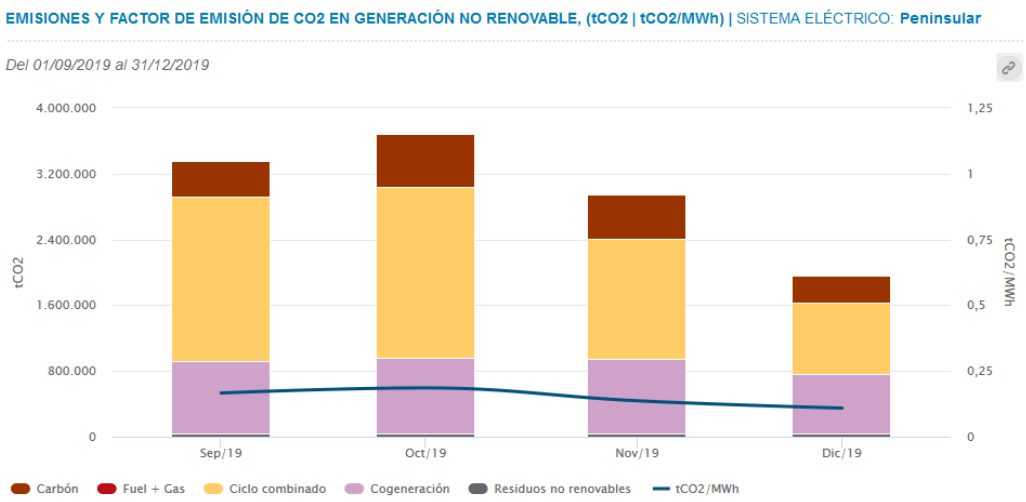

Figura 6.1. Emissions de CO2 al període de realització del treball

Les dades d'emissions en tones CO_2/MWh es detallen a la següent taula:

Setembre 2019	Octubre 2019	Novembre 2019	Desembre 2019
0,17	0,19	0,14	0,11

Taula 6.1. Dades d'emissions

Considerant que les hores dedicades a l'elaboració del treball s'han repartit uniformement entre els quatre mesos, realitzant una senzilla operació s'obtenen les emissions totals:

$$\frac{7{,}29}{1000} \times (0{,}17 + 0{,}19 + 0{,}14 + 0{,}11) = 0{,}0044469\ tones\ de\ CO_2 = 4{,}45\ kg\ de\ CO_2$$

(Equació 6.1)

L'altre aspecte ambiental que es pot considerar són les conseqüències que pot implicar l'aplicació d'aquest model a la realitat. En aquest cas, parlaríem d'un impacte ambiental positiu, ja que la predicció del consum elèctric pot ajudar a calcular quina és la demanda de forma més acurada, així subministrant la quantitat exacta que es necessita a cada moment, així reduint la predicció d'energia quan aquesta no sigui necessària.

A més, tenir un sistema de predicció d'aquest tipus és molt important en la situació actual a la qual es troben les fonts d'energia renovables, on fonts com pot ser l'eòlica necessiten una planificació per tal d'evitar problemes eventuals quan es produeix una variació, per exemple, en la velocitat del vent que limita la producció d'electricitat. D'aquesta manera es podria reduir el suport que es fa amb energies no renovables i, per tant, reduir les emissions.

7. Estudi econòmic

La principal despesa de la realització d'aquest Treball Fi de Grau ve provocada per la utilització dels equips informàtics necessaris per a la creació de la xarxa neuronal. Concretament, s'han fet servir dos ordinadors: un HP Envy 15k-201ns i un HP Stream 10ak-001ns. El cost d'adquisició del primer va ser de 999 euros a l'agost del 2015. Si s'ajustés aquest preu segons l'IPC a data de setembre de 2019 (quan es va començar a realitzar aquest treball), tenint en compte que s'ha registrat en aquest període un augment del IPC del 6,7% en productes informàtics [39], el cost seria de 1065,93 euros. L'altre ordinador es va comprar al mateix mes de setembre de 2019 i el seu preu va ser de 199 euros. Els programes informàtics emprats durant la realització del treball no han comportat cap cost, ja que Python és un programa amb llicència de codi obert.

Una altra despesa a tenir en compte és el consum elèctric dels ordinadors. La utilització d'aquests dos equips ha estat desigual: el primer ordinador, més potent, s'ha fet servir durant aproximadament el 80% del temps, i el segon durant el 20% del temps, bàsicament per a tasques de redacció de la memòria. El Treball Fi de Grau consta de 12 crèdits, el que equival a 30 hores de treball per crèdit, és a dir, un total de 360 hores de treball. Fent aquesta divisió, el primer ordinador ha estat funcionant durant 288 hores, i el segon durant 72 hores. En el primer cas, el consum aproximat és de 90 W segons el fabricant, i pel segon cas el consum és d'aproximadament 45 W. El preu mitjà del kWh (en realitat, no és un valor constant, sinó que va oscil·lant al llarg del temps) pels clients d'Endesa a l'any 2019 és de 0,12 €/kWh [40]. La despesa total en electricitat es fixa en 29160 W, o bé 29,160 kW, el que ens donaria un cost de 3,4992 €.

Per últim s'han de considerar les hores dedicades a totes les parts que conformen aquest Treball Fi de Grau. Com per a la despesa en electricitat, es consideren les 360 hores, amb una remuneració equivalent al mínim fixat per l'Escola per la realització de pràctiques acadèmiques externes, sent aquest cost de 8€ la hora, el que dona un cost total en mà d'obra de 2880€.

Així, el preu total quedaria fixat en 4148,43€. Aquesta xifra, però, no seria exacta si s'hagués de partir des de zero. En aquest cas, s'haurien de sumar tots els costos que ha comportat l'adquisició de les dades de consum elèctric que s'han anat recopilant durant quatre anys i mig, que haurien de comportar les despeses tant en la compra del sistema encarregat de

l'adquisició de les dades com la mà d'obra dedicada a vetllar i controlar periòdicament el correcte funcionament del sistema i que, per tant, pogués garantir que els valors de consum registrat són fidels a la realitat. A continuació es mostra una taula, a mode de resum, que conté tots els costos associats a la realització del treball:

Concepte	Cost
Ordinador 1	1065,93 €
Ordinador 2	199 €
Programari informàtic (Python i biblioteques)	0 €
Consum elèctric dels ordinadors	3,4992 €
Mà d'obra	2880 €
TOTAL	**4148,83 €**

Taula 7.1. Cost de la realització del treball

Si es posa la vista en quin és l'impacte econòmic que pot comportar l'aplicació d'aquest projecte a un entorn real, clarament aquest seria el d'una reducció de costos, ja que el desenvolupament d'un sistema capaç de predir el consum elèctric pot ajudar a gestionar la xarxa elèctrica d'una forma molt més eficient, distribuint l'electricitat adaptada a les necessitats dels usuaris en un moment determinat, i no fent-ho de forma que sempre arribi la mateixa quantitat sense tenir informació de si serà excessiva o insuficient.

Conclusions

Les xarxes neuronals són una eina molt potent que pot servir per realitzar tasques molt complexes com pot ser realitzar una predicció del consum futur, cosa que pot semblar d'una elevada dificultat i pràcticament irreal amb el nombre de dades sobre la demanda d'energia de la qual es disposava a l'inici de l'estudi. A més, les xarxes neuronals presenten un gran avantatge respecte a altres tècniques per predir la demanda elèctrica, com poden ser els models teòrics, ja que tenen la capacitat d'adaptar-se a dades i situacions molt concretes, com poden ser els hàbits de consum d'un habitatge, mentre que la creació d'un model matemàtic que es pugui ajustar a aquestes circumstàncies concretes seria molt complex. Tot i aquesta variabilitat, les dades estudiades presenten una elevada variabilitat, i una gran quantitat de pics de consum que no fan gens fàcil la tasca.

S'ha observat que amb el nombre suficient de dades i paràmetres s'ha aconseguit una predicció fiable i en línies generals ben ajustada a la realitat. A més, s'han predit amb fidelitat les tendències. No obstant això, el consum elèctric d'un habitatge depèn d'un nombre molt elevat de factors que fan que certs aspectes de la demanda elèctrica siguin pràcticament impredictibles. Ha quedat demostrat que comptar amb un nombre elevat de dades és essencial per fer que els resultats siguin el més acurats possible. Ha estat per això pel que la primera prova, on només s'actualitzava la predicció una vegada al dia les prediccions no eren gens reals. Aquesta prova però, ha servit per establir els criteris de comparació i quines variables són d'interès per a l'estudi. També ha resultat útil com a prova per a després realitzar la prova número 2, on s'ha definit que la predicció s'actualitzi cada cinc minuts, amb el que s'ha comprovat que els resultats milloren significativament. Això ha suposat el punt de partida per a realitzar el procés de selecció de les variables que tenen un impacte positiu en les prediccions. Les conclusions més importants a les quals s'ha arribat són les següents:

- Per a realitzar un anàlisi que s'executi el més ràpid possible i sense cap tipus de modificació dels paràmetres propis de la xarxa neuronal, l'optimitzador més adequat dels triats per l'estudi és *Adam,* que porta en general a la solució correcta amb menor dificultat.

- El nombre de neurones, en el cas de la creació d'una xarxa neuronal amb una única capa, és determinant per a assolir la millor predicció possible. A més, un augment indiscriminat d'aquest nombre pot portar a lleugerament pitjors resultats. Amb una

configuració de 500 neurones s'arriba al mínim error.

- L'addició d'una segona capa de neurones millora la precisió de la xarxa en comparació a quan només es té una, però el nombre de neurones que es tingui no sembla ser massa significatiu si es manté dins el rang entre 50 i 350 neurones. A partir de 400 neurones la xarxa es torna massa complexa, esdevé incapaç d'actualitzar els pesos i el resultat és una predicció constant. Això demostra un dels problemes de les funcions d'activació *ReLU*: la incapacitat per activar-se quan la xarxa és molt profunda, un fenomen conegut com a *dying ReLU.*

- La incorporació del dia de la setmana presenta un impacte positiu en les prediccions, que no és tan significatiu en el cas de la inclusió de les dades del mes.

- La forma d'incorporar les dades addicionals és molt important i té gran efecte en els resultats: la incorporació de les mateixes dades, però d'una manera diferent, en el cas de les dades meteorològiques, dona lloc a una variació en l'error semblant a la que hi ha quan s'inclouen les dades del dia de la setmana. També s'observa les implicacions que poden tenir aquestes modificacions: en la segona prova, al haver reduït el nombre de mostres que es tenien, el fenomen de la *dying ReLU* es va començar a produir amb menys neurones del què es produïa abans, havent de reduir la taxa d'aprenentatge per solucionar-ho.

El treball realitzat, però, es pot continuar a futurs estudis avaluant altres aspectes relacionats amb l'estructura de les dades: en les dues proves s'ha seguit el que van demostrar els estudis d'autocorrelació realitzats a l'inici de l'anàlisi: la periodicitat més marcada es produeix cada 24 hores, per tant, es va triar fer que amb 24 hores d'informació es prediguin les 24 següents. No obstant això, s'observa que també hi ha una correlació elevada (tot i que no tant) a les 48 hores, per tant, és d'especial interès analitzar que succeeix quan, per exemple, s'empren les dades de 48 hores per a predir les 24 hores de consum elèctric següents.

A més, les prediccions obtingudes podrien ser de més precisió si es tinguessin en comptes altres factors i dades de les quals no s'ha pogut disposar, per exemple, quins dies eren festius, fet que clarament té una influència en la demanda d'electricitat. Després d'haver analitzat els resultats obtinguts, es pot concloure que s'han assolit els objectius marcats a l'inici del treball.

Agraïments

Vull agrair a Ramon Costa Castelló tota l'ajuda, suport i consells que m'ha ofert durant la realització d'aquest Treball Fi de Grau.

Finalment, donar les gràcies als meus pares i a la meva germana per estar al meu costat i donar-me suport: gràcies a vosaltres he pogut arribar fins aquí.

Bibliografia

Referències bibliogràfiques

[1] DJURIS, J., MEDAREVIC, D., KRSTIC, M., VASILJEVIC, I., ALEKSIC, I., IBRIC, S. (2012). Schematic drawing of multilayer perceptron neural networks [il·lustració]. A: Design Space Approach in Optimization of Fluid Bed Granulation and Tablets Compression Process. *The Scientific World Journal*. Hindawi Publishing Corporation. 2012. Article ID 185085, p. 2. ISSN 1537-744X.

[2] HEATON, J. *The nombre of hidden layers.* 2017. [Consulta: 30 de novembre 2019]. Disponible a: < https://www.heatonresearch.com/2017/06/01/hidden-layers.html>

[3] COOPER, M. J. Sigmoid Activation Function. [il·lustració]. A: A Deep Learning Prediction Model for Mortgage Default A Deep Learning Prediction Model for Mortgage Default. Treball Fi de Màster, Universitat de Bristol, 2018, p. 15. [Consulta: 18 de novembre 2019]. Disponible a: < https://www.researchgate.net/publication/330303425_A_Deep_Learning_Prediction_Model_for_Mortgage_Default_A_Deep_Learning_Prediction_Model_for_Mortgage_Default>

[4] COOPER, M. J. ReLU Activation Function. [il·lustració]. A: A Deep Learning Prediction Model for Mortgage Default A Deep Learning Prediction Model for Mortgage Default. Treball Fi de Màster, Universitat de Bristol, 2018, p. 17. [Consulta: 18 de novembre 2019]. Disponible a: < https://www.researchgate.net/publication/330303425_A_Deep_Learning_Prediction_Model_for_Mortgage_Default_A_Deep_Learning_Prediction_Model_for_Mortgage_Default>

[5] GOOGLE. *Reducción de la pérdida: Tasa de aprendizaje.* [Consulta: 4 de gener 2020]. Disponible a: < https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate?hl=es-419>

[6] SCITKIT-LEARN. *Overfitting vs. Underfitting.* 2014. [Consulta: 4 de gener 2020]. Disponible a: < https://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html>

[7] NOVAK, G.S. Essay: Artificial Intelligence. *Academic Press Dictionary of Science and Technology.* 1992, p. 160.

[8] MILLER, C.M. *Ada Lovelace: A gifted mathematician who is now recognized as the first computer programmer*. The New York Times. Nova York, 2018. [Consulta: 30 de novembre 2019]. Disponible a : <https://www.nytimes.com/interactive/2018/obituaries/overlooked-ada-lovelace.html>

[9] MUGGLETON, S. *Alan Turing and the development of Artificial Intelligence.* [En línia]. Londres, 2012. [Consulta: 1 desembre 2019]. Disponible a: < https://www.lirmm.fr/ecai2012/images/stories/ecai_doc/pdf/Turing/Muggleton_ECAI.pdf >

[10] MCCARTHY, J. History of Lisp. A: *History of programming languages.* [en línia]. Los Angeles, 1978. [Consulta: 1 de desembre 2019]. Disponible a : < http://jmc.stanford.edu/articles/lisp/lisp.pdf>

[11] ANYOHA, R.. *The history of artificial intelligence*. Harvard: Harvard University, 2017. p. 173-185. [Consulta: 1 de desembre 2019]. Disponible a: < http://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>

[12] ZENG, X. *Lecture 4 – Overview and brief history of AI.* s.d. [Consulta: 1 de desembre 2019]. Disponible a: < http://syllabus.cs.manchester.ac.uk/ugt/COMP14112/lectures/large/lecture4.ppt>

[13] BISHOP, J. The structure of Rosenblatt's Perceptron. A: History and philosophy of neural networks [en línia]. París, Encyclopedia of Life Support Systems, 2015. Capítol 2, p. 1-74.

[14] BEEMAN, D. *Some specific models of artificial neural nets* [en línia]. Colorado: Universitat de Colorado, 2004. [Consulta: 3 de desembre 2019]. Disponible a: < http://ecee.colorado.edu/~ecen4831/lectures/NNet2.html>

[15] BISHOP, J. The structure of Rosenblatt's Perceptron [il·lustració]. A: History and philosophy of neural networks [en línia]. París, Encyclopedia of Life Support Systems, 2015. Capítol 2, p. 18.

[16] WERBOS, P. J. Beyond regression : new tools for prediction and analysis in the behavioral sciences. Tesi doctoral, Universitat de Harvard, 1974. [Consulta: 3 de desembre 2019]. Disponible a: < https://www.researchgate.net/publication/35657389_Beyond_regression_new_tools_for_prediction_and_analysis_in_the_behavioral_sciences>

[17] NVIDIA. *Long Short-Term Memory [LSTM].* NVIDIA Corporation, s.d. [Consulta: 4 de desembre 2019]. Disponible a: < https://developer.nvidia.com/discover/lstm>

[18] ROBERTS, E. Neural Networks. History: The 1940's to 1970's. Universitat de Stanford. [Consulta: 2 de desembre 2019]. Disponible a: < https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html>

[19] CHOLLET, F. A 3D timeseries data tensor [il·lustració].A: Deep Learning With Python. Nova York: Manning Publications, 2018, p.35. ISBN 9781617294433.

[20] SCIPY. *Frequently Asked Questions.* [Consulta: 30 de setembre 2019]. Disponible a: < https://www.scipy.org/scipylib/faq.html>

[21] NAU, B. *Three types of forecasts: estimation, validation, and the future.* Universitat de Duke. [Consulta: 13 d'octubre 2019]. Disponible a: <http://people.duke.edu/~rnau/three.htm>

[22] AL-GBURI, M. "Can someone recommend what is the best percent of divided the training data and testing data in neural network 75:25 or 80:20 or 90:10 ?" [Pregunta a un fòrum]. ResearchGate [en línia]. 12 de desembre de 2014. [Consulta: 6 d'octubre 2019]. Disponible a: <https://www.researchgate.net/post/can_someone_recommend_what_is_the_best_percent_of_divided_the_training_data_and_testing_data_in_neural_network_7525_or_8020_or_9010>

[23] RYSER-WELCH, P. "Is there an ideal ratio between a training set and validation set? Which trade-off would you suggest?" [Pregunta a un fòrum]. ResearchGate [en línia]. 3 de març de 2016. [Consulta: 6 d'octubre 2019]. Disponible a: <https://www.researchgate.net/post/Is_there_an_ideal_ratio_between_a_training_set_and_validation_set_Which_trade-off_would_you_suggest>

[24] STANFORD UNIVERSITY. *Optimization: Stochastic Gradient Descent.* Universitat de Stanford. [Consulta: 14 de novembre 2019]. Disponible a: <http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/>

[25] AKSOY, S., HARALICK, R. M. Feature Normalization and Likelihood-based Similarity Measures for Image Retrieval. *Pattern Recognition Letters*. Elsevier, 2001. ISSN 0167-8655.

[26] GOLDEN, B., LE GRAND, B., MYTTENAERE, A, ROSSI, F. Mean Absolute Percentage Error for regression models. A: *Advances in artificial neural networks,machine learning and computational intelligence - Selected papers from the 23rd European Symposiumon Artificial Neural Networks (ESANN 2015)*.

Neurocomputing, Elsevier, 2016, 192, p.38 - 48.

[27] NAU, B. *What's the bottom line? How to compare models.* Universitat de Duke. [Consulta: 10 de novembre 2019]. Disponible a: <https://people.duke.edu/~rnau/compare.htm>

[28] RAMACHANDRAN, P. *Advanced Training Techinques.* Universitat d'Illinois. [Consulta: 9 de novembre 2019]. Disponible a: <http://slazebni.cs.illinois.edu/spring17/lec05_advanced_training.pdf>

[29] KINGMA, D. P., LEI BA, J. ADAM: A method for stochastic optimization. A: *3rd International Conference on Learning Representations*. San Diego, 2015, s.p. [Consulta: 16 de novembre de 2019]. Disponible a: <https://arxiv.org/pdf/1412.6980.pdf>

[30] KESHAR, N., MUDIGERE, D., NOCEDAL, J., SMELYANSIY, M., TANG, P.. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. A: *5th International Conference on Learning Representations.* Toulon, 2017, s.p. [Consulta: 29 de novembre de 2019. Disponible a: <https://arxiv.org/pdf/1609.04836.pdf>

[31] CAO, W., LIU, L., PU, C., SAHIN, S., WEI, W., WU, Y., ZHANG, Q.. A Comparative Measurement Study of Deep Learning as a Service Framework. A: *IEEE Transactions on Services Computing*. 2019, s.p. [Consulta: 30 de novembre 2019]. Disponible a: <https://arxiv.org/pdf/1810.12210.pdf>

[32] ZHANG, Y., QU, H., CHEN, C., METAXAS, D. Taming the Noisy Gradient: Train Deep Neural Networks with Small Batch Sizes. A: *Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. Macau, 2019, p. 4348-4354. [Consulta: 24 de novembre 2019].Disponible a:<https://www.ijcai.org/proceedings/2019/0604.pdf>

[33] SHEELA, K., DEEPA, S N. Review on Methods to Fix Number of Hidden Neurons in Neural Networks. A: *Mathematical Problems in Engineering*. Hindawi Publishing Corporation: 2013, s.p.. ISSN 1563-5147. [Consulta: 23 de novembre 2019]. Disponible a: <https://www.researchgate.net/publication/258393467_Review_on_Methods_to_Fix_Number_of_Hidden_Neurons_in_Neural_Networks>

[34] KLEINBERG, R., LI, Y., YUAN, Y. (2018). *An Alternative View: When Does SGD Escape Local Minima?. Proceedings of the 35th International Conference on Machine Learning*. [en línia]. Estocolm: PMLR, 2019, Volum 80, p. 2698-2707. [Consulta: 30 de novembre 2019]. Disponible a: <https://arxiv.org/pdf/1802.06175.pdf>

[35] O'LEARY, D., YOUSEFZADEH, R. *Refining the Structure of Neural Networks Using Matrix Conditioning.* 2019. [Consulta: 4 de desembre 2019]. Disponible a.: <https://arxiv.org/pdf/1908.02400.pdf>

[36] JOHNSON, J., KARPATHY, A. *CS231n Convolutional Neural Networks for Visual Recognition.* [Consulta: 23 de desembre 2019]. Disponible a: <http://cs231n.github.io/neural-networks-1/>

[37] KARNIADAKIS, G., LU, L.., SHIN, Y., SU, Y. *Dying ReLU and Initialization: Theory and Numerical Examples.* 2019. [Consulta: 26 de desembre 2019]. Disponible a: <https://arxiv.org/pdf/1903.06733.pdf>

[38] RED ELÉCTRICA DE ESPAÑA. *Generación.* [Consulta: 25 de desembre 2019]. Disponible a: <https://www.ree.es/es/datos/generacion>

[39] INSTITUTO NACIONAL DE ESTADÍSTICA. *Cálculo de variaciones del Indice de Precios de Consumo (sistema IPC base 2016).* [Consulta: 25 de desembre 2019]. Disponible a: <https://www.ine.es/varipc/verVariaciones.do?idmesini=9&anyoini=2015&idmesfin=9&anyofin=2019&ntipo=4&enviar=Calcular>

[40] ENDESA. *Tarifas de luz.* [Consulta: 25 de desembre 2019]. Disponible a: <https://www.endesaclientes.com/catalogo/luz.html>

Bibliografia complementària

CHOLLET, F. *Deep Learning With Python*. Nova York: Manning Publications, 2018. ISBN 9781617294433.

Annex

Fitxers

Adjunt a aquesta memòria, es facilita una carpeta que conté tots els arxius utilitzats per a la creació de la xarxa neuronal, tant com per a la xarxa en sí com per al processat de les dades. El codi de cadascun dels arxius està comentat per tal d'identificar-ne les parts bàsiques del mateix; també s'indica què s'ha de canviar per realitzar les diferents proves, en els casos als quals un mateix programa s'ha fet servir en més d'una prova. A continuació es presenta una llista dels arxius presents a la carpeta, en ordre de creació i d'utilització. Per a poder executar aquesta carpeta, cal tenir instal·lats a l'ordinador, a més del programa Python, les biblioteques Keras, matplotlib, NumPy, SciPy i Tensorflow.

- *consum_2015_2018.csv* i *consum_01_2019_06_2019.txt* contenen les dades inicials corresponents al període comprès entre gener de 2015 i desembre de 2018, i entre gener de 2019 i juny de 2019, respectivament.

- *uneix_dades.py* tracta els dos arxius esmentats anteriorment i els guarda units en un format adient.

- *dades_unides.txt* conté les dades processades per l'arxiu anterior.

- *passar_a_tensor3D.py* recupera les dades de l'arxiu anterior i les transforma al format de tensor 3D.

- *dades_tensor3D.npy* emmagatzema les dades tractades per l'arxiu previ.

- *delma_dades.py* realitza el procés de delmació i interpolació de les dades.

- *dades_delmades.npy* i *dades_interpolades.npy* contenen les dades resultants del procés de delmació i d'interpolació, respectivament,

- *dades_graficades.py* crea un gràfic comparant les dades delmades i interpolades contingudes en els arxius anteriors.

- *xarxa_dades_normalitzades.py* conté la xarxa neuronal emprada a les proves 1.1 i 1.2.. *xarxa_dades_normalitzades.py*, per la seva part, conté la xarxa de les proves

1.3 i 1.4. Per canviar l'optimitzador únicament cal modificar el nom del mateix pel que es vulgui utilitzar.

- *xarxa_una capa oculta.py* és la xarxa neuronal utilitzada a les proves 2.1 i 2.2. Com al cas anterior, els canvis necessaris per executar les diferents proves estan indicats al mateix codi.

- *xarxa_dues capes ocultes.py* s'ha fet servir a les proves 2.3, 2.4 i 2.5. En aquest cas, el propi fitxer conté anotacions amb el què s'ha de canviar per realitzar cadascuna de les proves.

- *xarxa_dues capes ocultes_amb flag.py* i *xarxa_dues capes ocultes_eliminant dies sense dades temps.py* són les xarxes neuronals emprades a les proves 2.6.1 i 2.6.2, respectivament.

Gràfics addicionals

Durant la realització de les proves 2.2 i 2.3 s'han realitzat proves successives per determinar el millor nombre de neurones per capes, que han comportat la generació d'un gran nombre de gràfics referents a cadascuna d'aquestes. Al nucli de la memòria únicament es mostra un resum de com evoluciona l'RMSE segons el nombre de neurones, ja que la inclusió de la resta de gràfics i dades pot fer més complicada la lectura i interpretació dels resultats i no són essencials per comprendre la finalitat d'aquesta etapa ni el per què es tria un nombre de neurones concret: no obstant això, permeten visualitzar amb més detall com milloren els resultats amb l'increment del nombre de neurones. A continuació es mostra els gràfics de la *loss function,* RMSE, una visió general de les prediccions i la selecció aleatòria de 6 mostres per a cadascuna de les proves.

La numeració de cada una de les subproves segueix el criteri anteriorment: la 2.2.1 correspon a la que conté 50 neurones en una única capa, la 2.2.2 a la que té 100 neurones en una única capa i així successivament. Primer de tot, es mostra una taula que resum el valor de la *loss function,* del RMSE i les iteracions necessàries per a totes les subproves realitzades dins de les proves 2.2 i 2.3.

- **A. Prova 2.2**

Nombre de neurones	Test loss	Test RMSE	Iteracions
50	0.00327	0.05771	29
100	0.00323	0.05730	25
150	0.00321	0.05724	19
200	0.00322	0.05705	23
250	0.00318	0.05695	24
300	0.00320	0.05684	30
350	0.00321	0.05712	14
400	0.00320	0.05687	25
450	0.00318	0.05675	30
500	0.00318	0.05667	30
550	0.00318	0.05686	23
600	0.00318	0.05691	20
650	0.00319	0.05675	27
700	0.00319	0.05696	18

Taula A.1. Dades del conjunt de test per a la prova 2.2

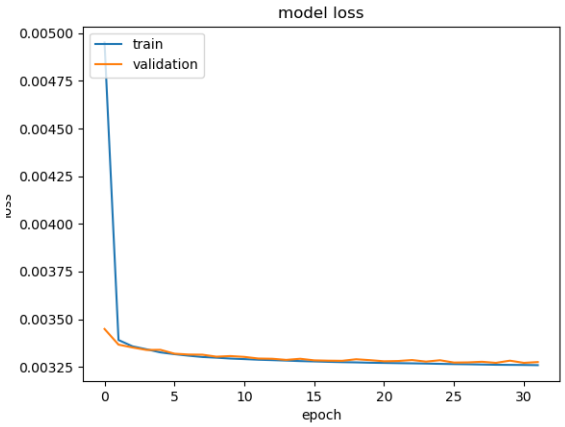

Figura A.1. Loss de la prova 2.2.1

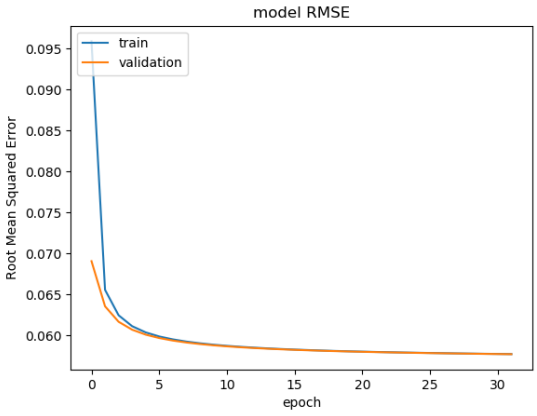

Figura A.2. RMSE de la prova 2.2.1

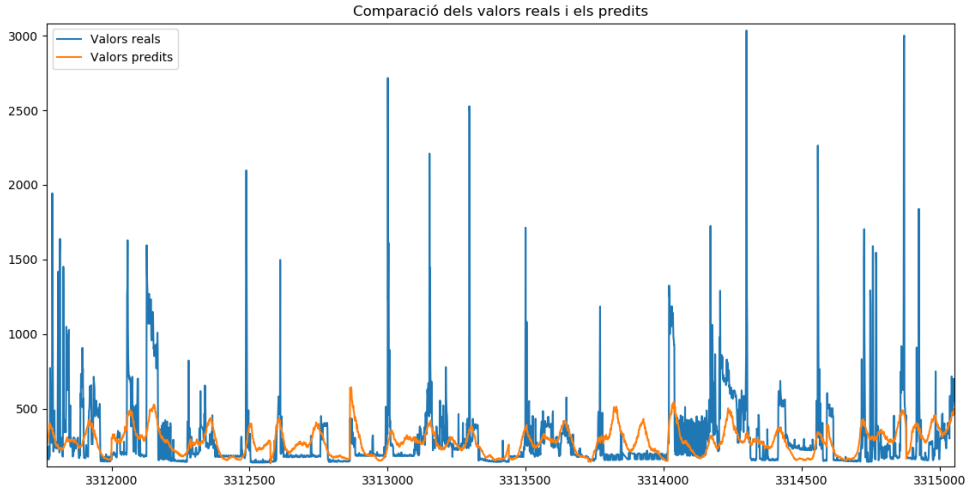

Figura A.3. Prediccions de la prova 2.2.1

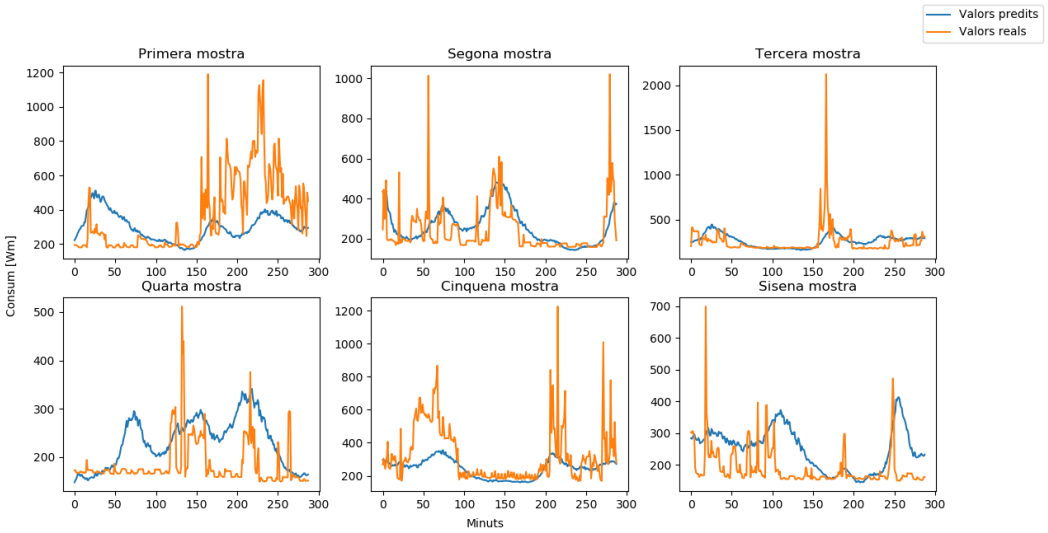

Figura A.4. 6 mostres de la prova 2.2.1

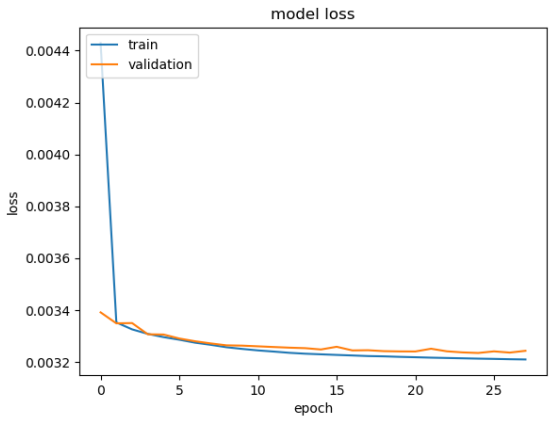

Figura A.5. Loss de la prova 2.2.2

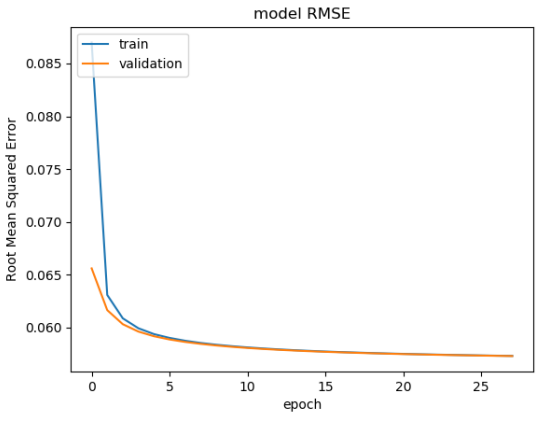

Figura A.6. RMSE de la prova 2.2.2

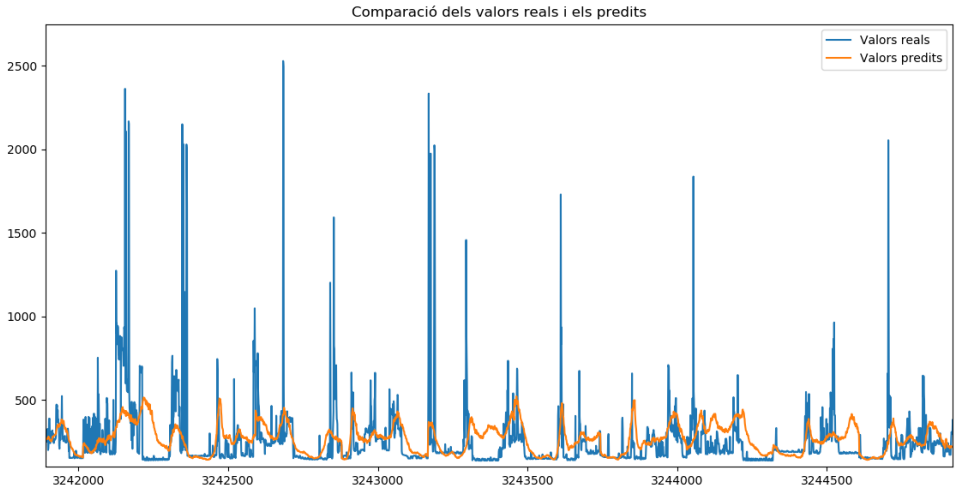

Figura A.7. Prediccions de la prova 2.2.2

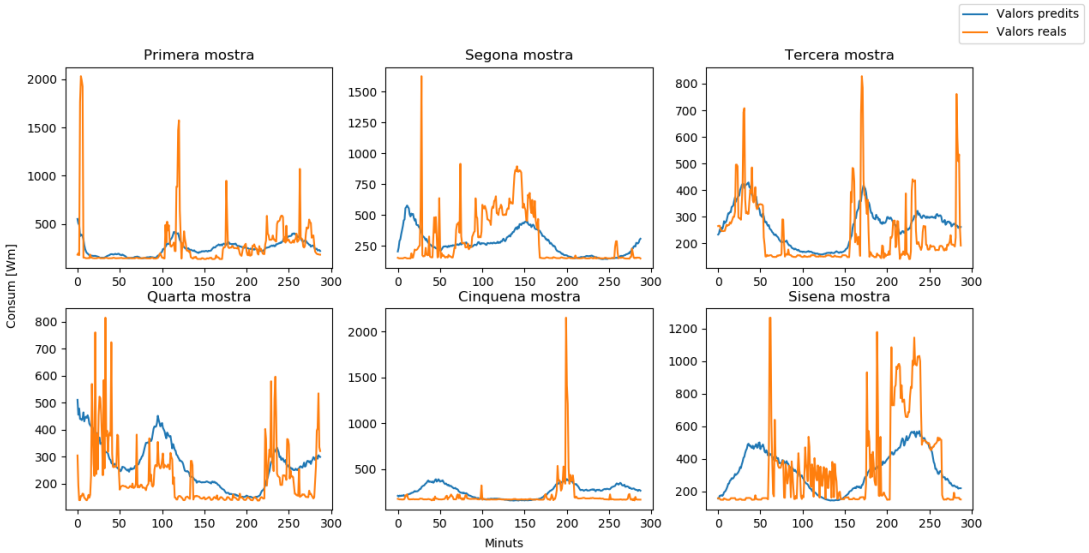

Figura A.8. 6 mostres de la prova 2.2.2

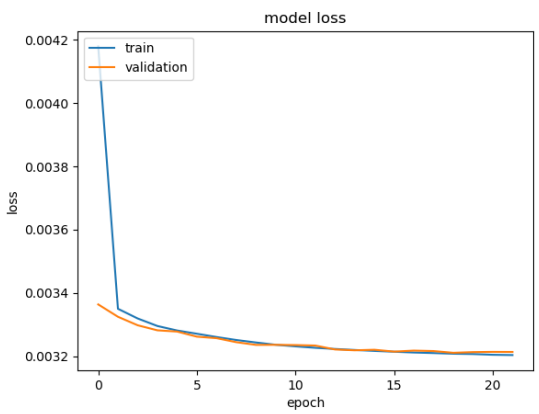

Figura A.9. Loss de la prova 2.2.3

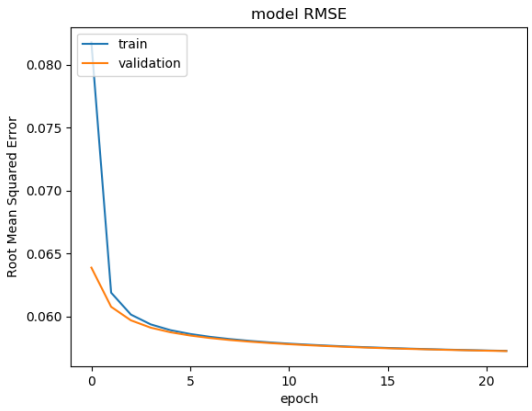

Figura A.10. RMSE de la prova 2.2.3

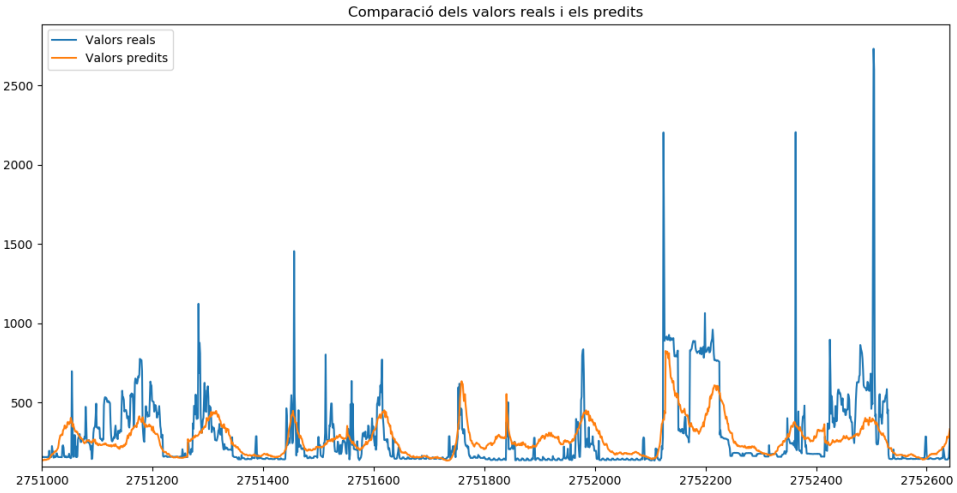

Figura A.11. Prediccions de la prova 2.2.3

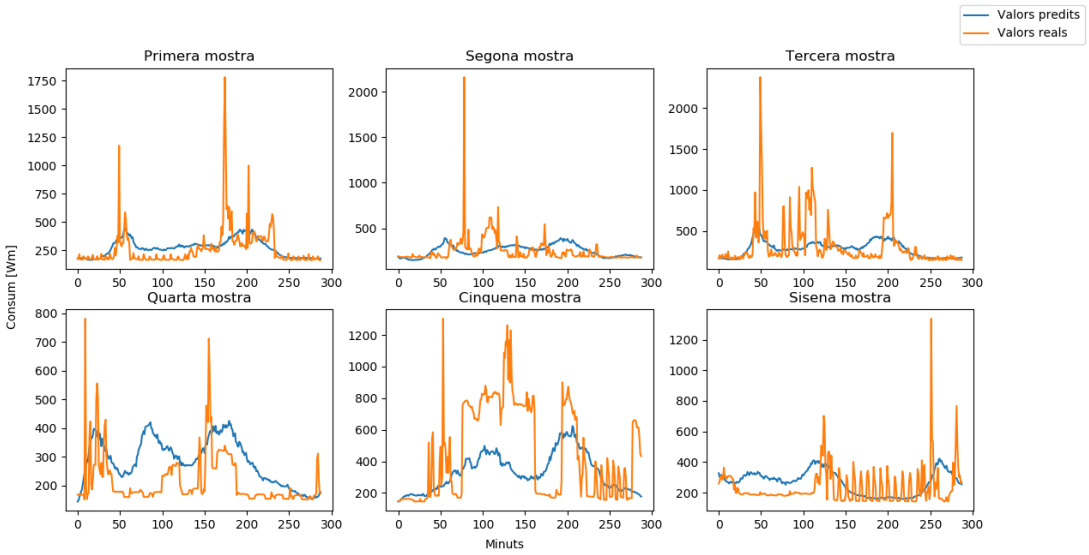

Figura A.12. 6 mostres de la prova 2.2.3

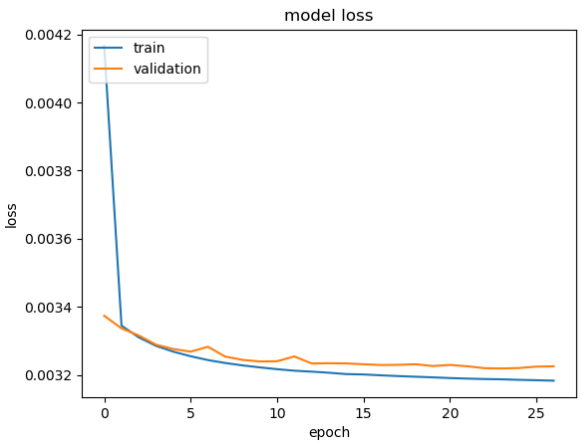

Figura A.13. Loss de la prova 2.2.4

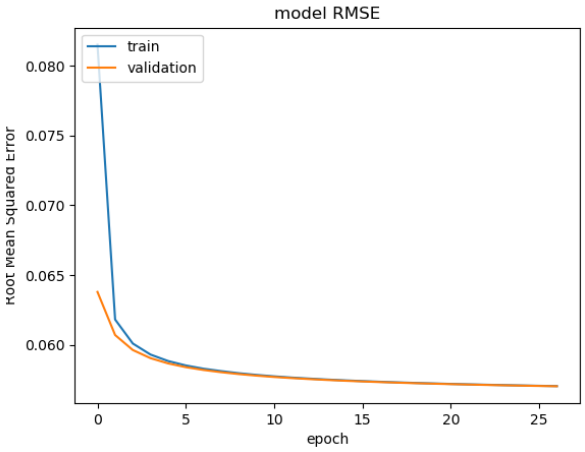

Figura A.14. RMSE de la prova 2.2.4

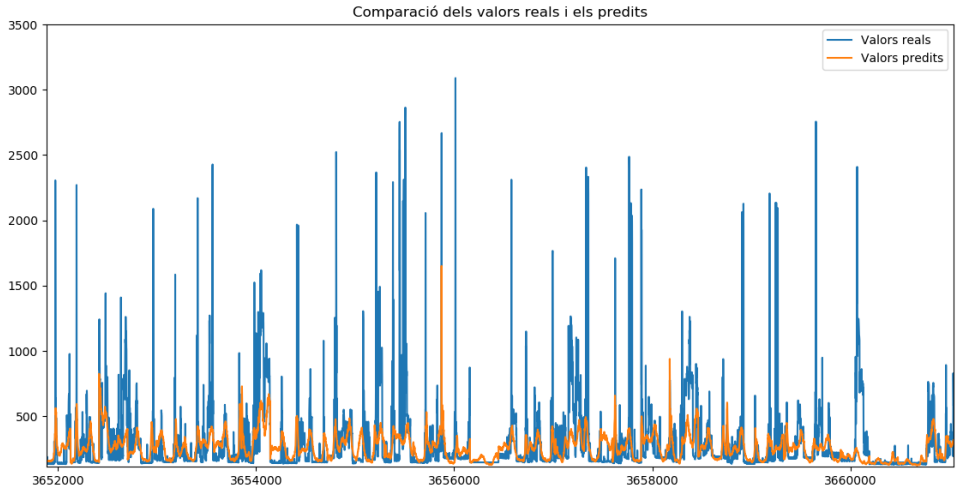

Figura A.15. Prediccions de la prova 2.2.4

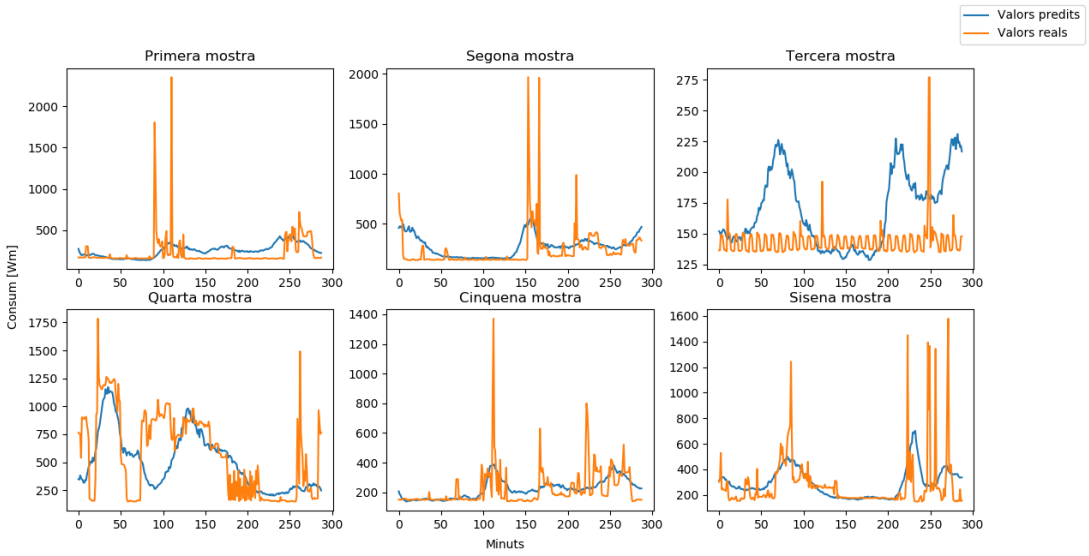

Figura A.16. 6 mostres de la prova 2.2.4

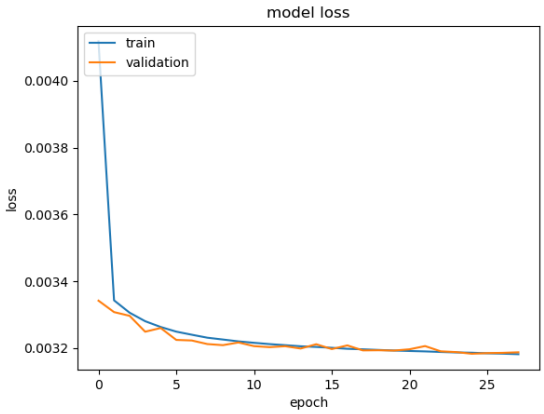

Figura A.17. Loss de la prova 2.2.5

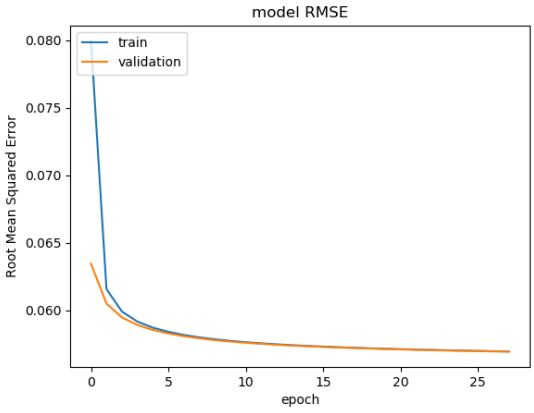

Figura A.18. RMSE de la prova 2.2.5

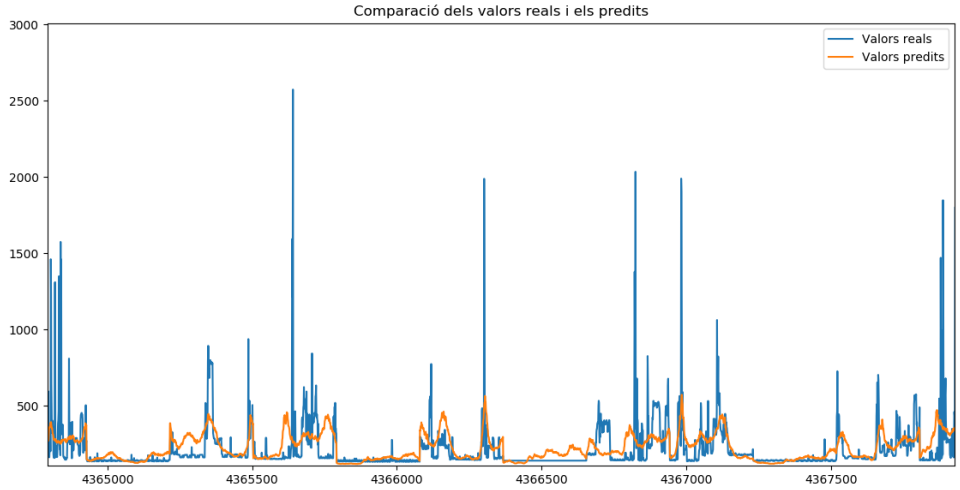

Figura A.19. Prediccions de la prova 2.2.5

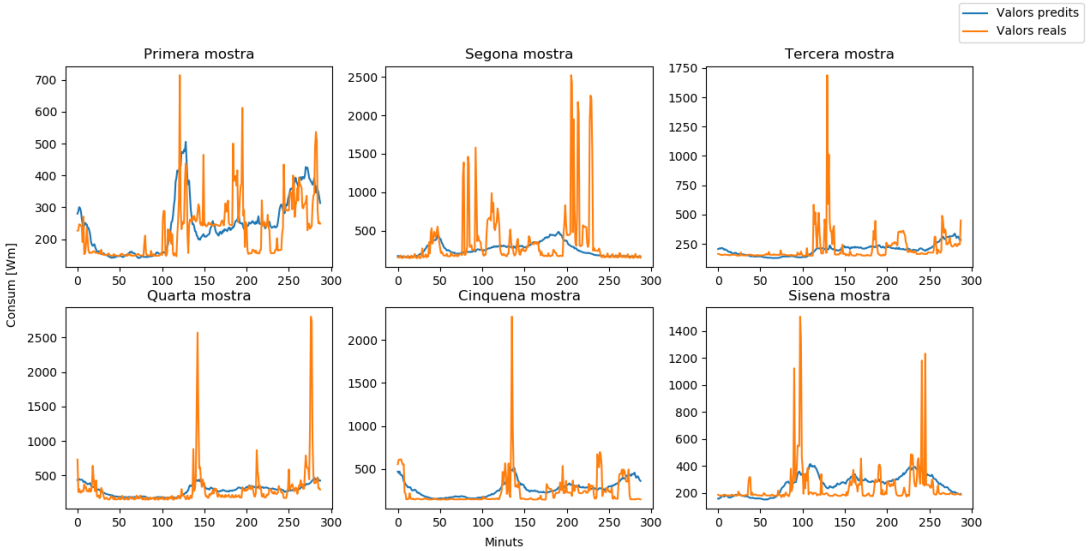

Figura A.20. 6 mostres de la prova 2.2.5

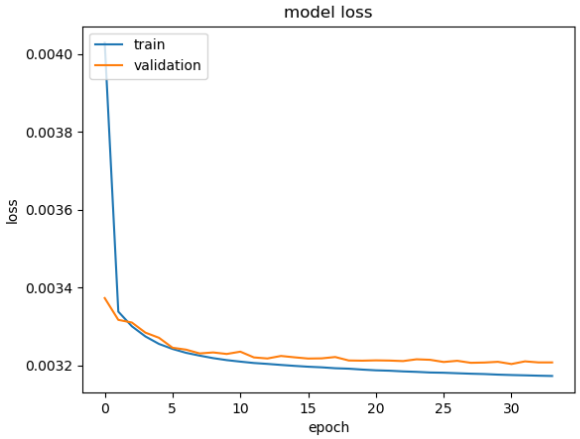

Figura A.21. Loss de la prova 2.2.6

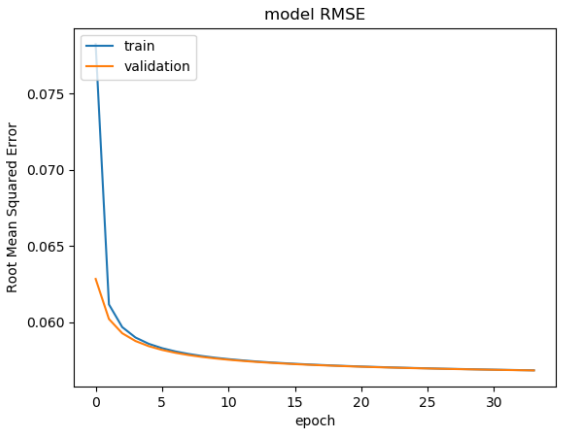

Figura A.22. RMSE de la prova 2.2.6

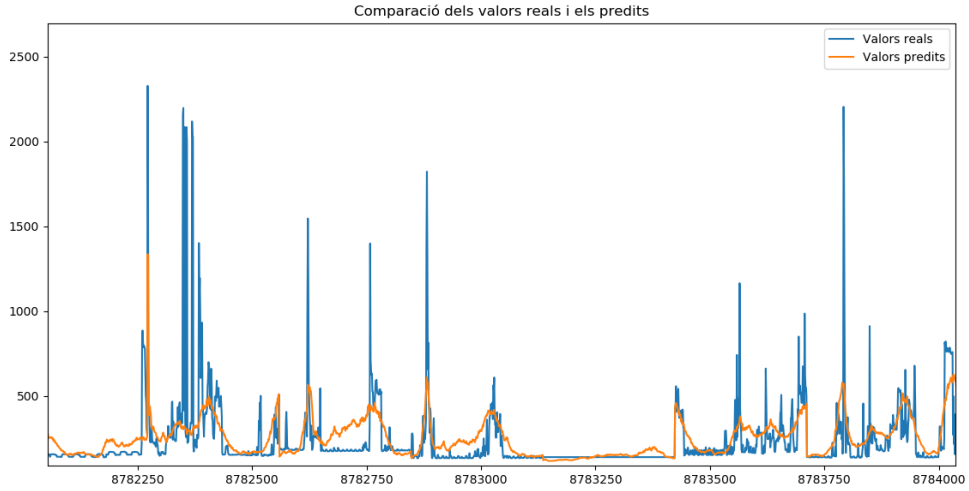

Figura A.23. Prediccions de la prova 2.2.6

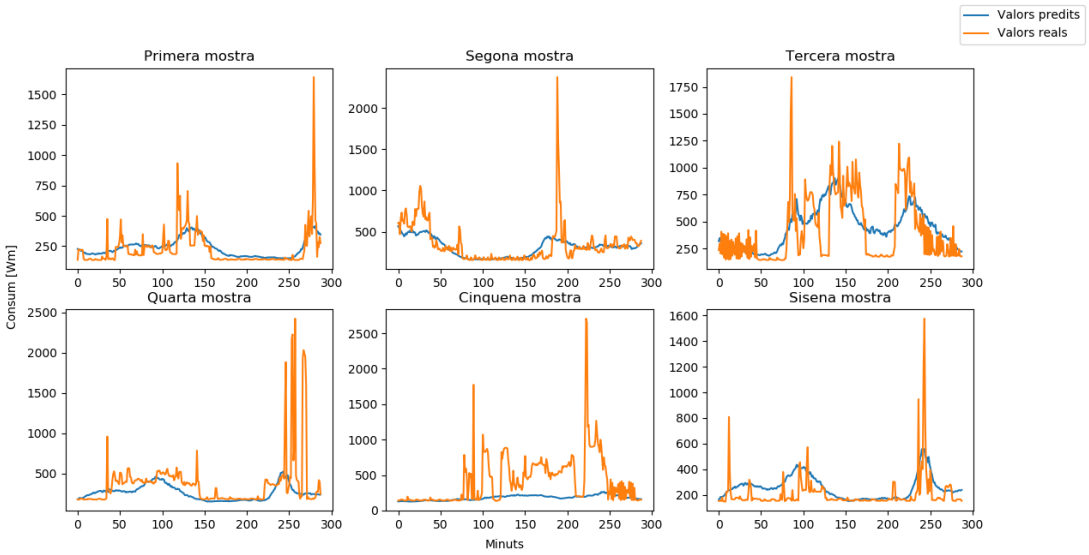

Figura A.24. 6 mostres de la prova 2.2.6

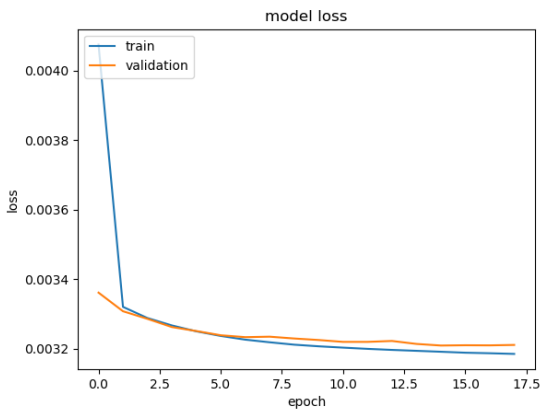

Figura A.25. Loss de la prova 2.2.7

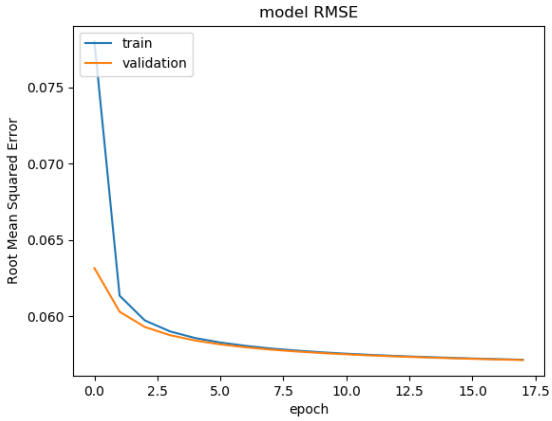

Figura A.26. RMSE de la prova 2.2.7

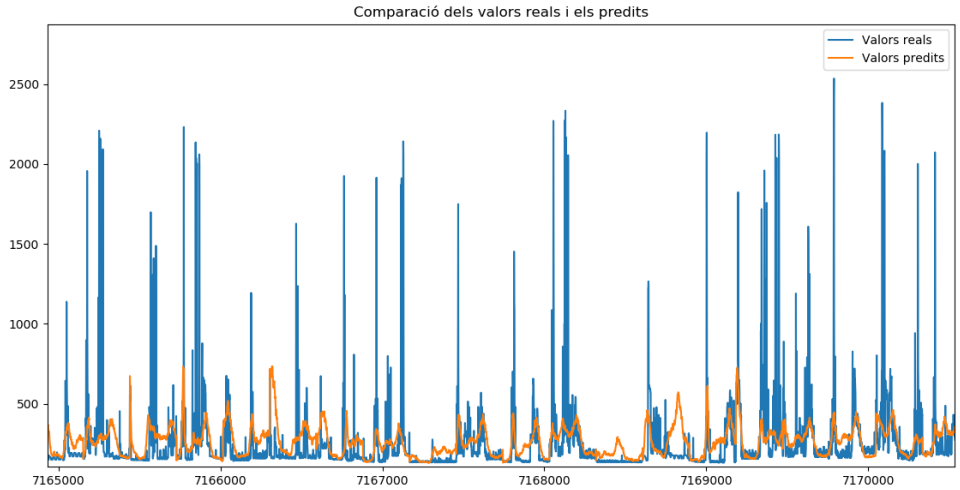

Figura A.27. Prediccions de la prova 2.2.7

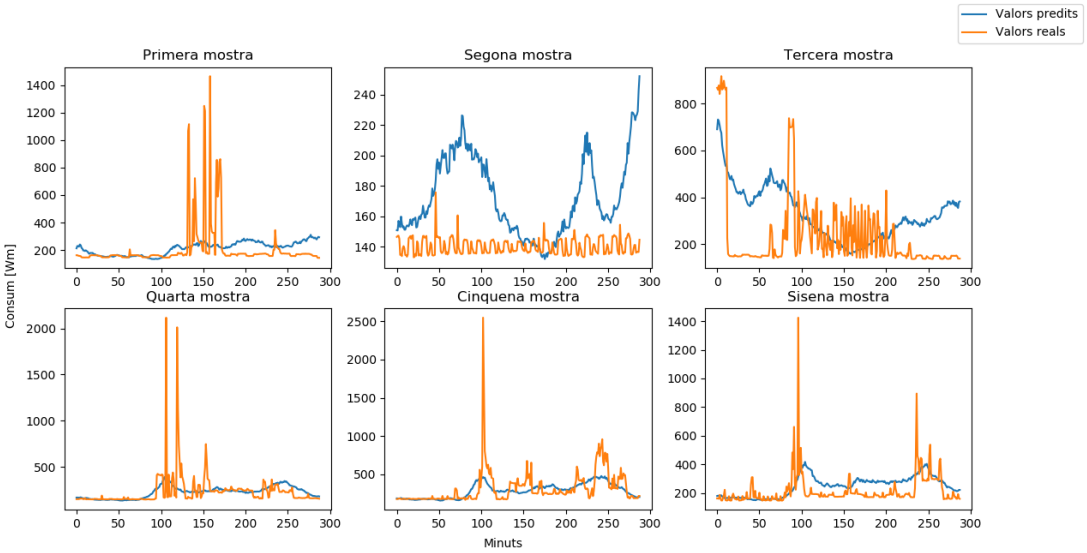

Figura A.28. 6 mostres de la prova 2.2.7

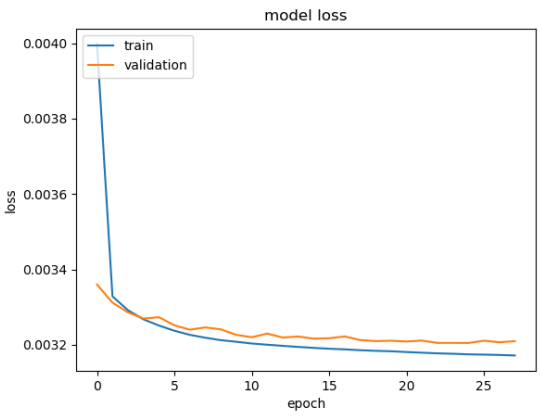

Figura A.29. Loss de la prova 2.2.8

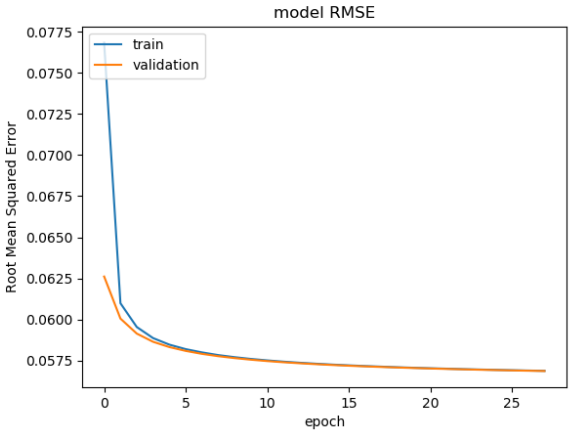

Figura A.30. RMSE de la prova 2.2.8

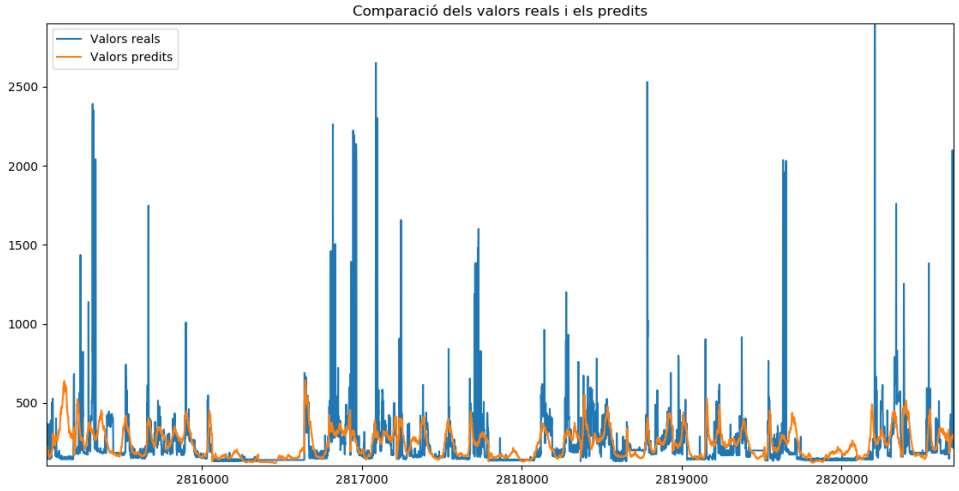

Figura A.31. Prediccions de la prova 2.2.8

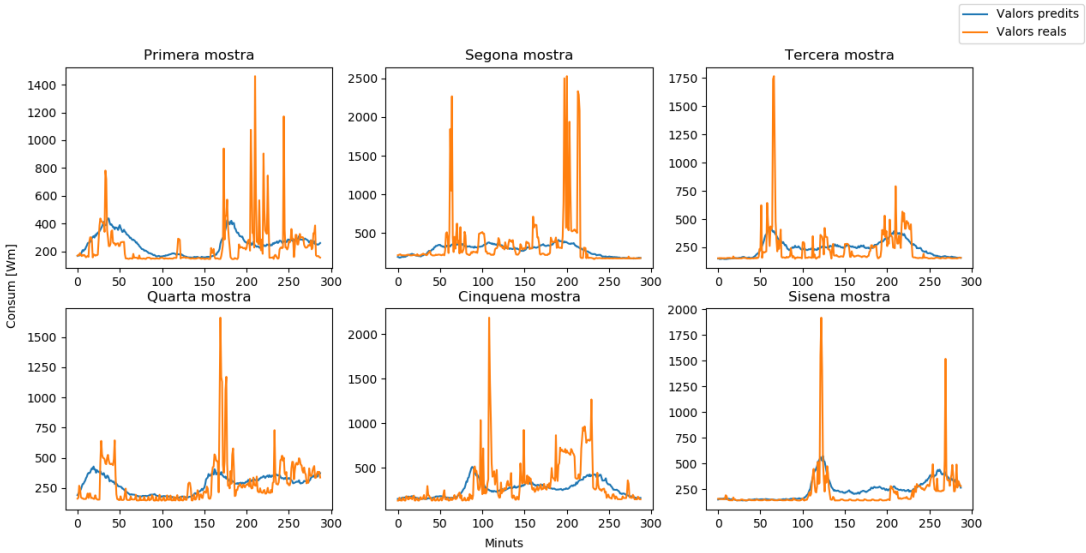

Figura A.32. 6 mostres de la prova 2.2.8

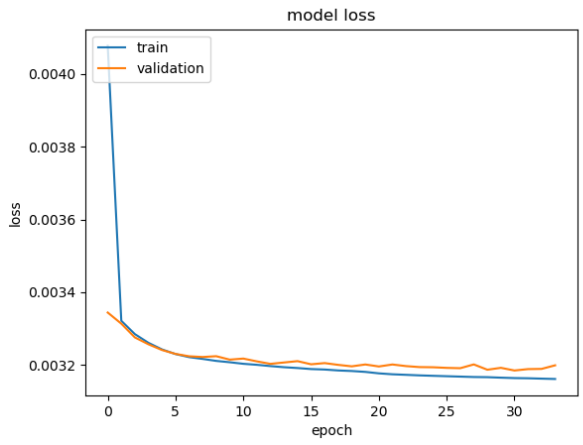

Figura A.33. Loss de la prova 2.2.9

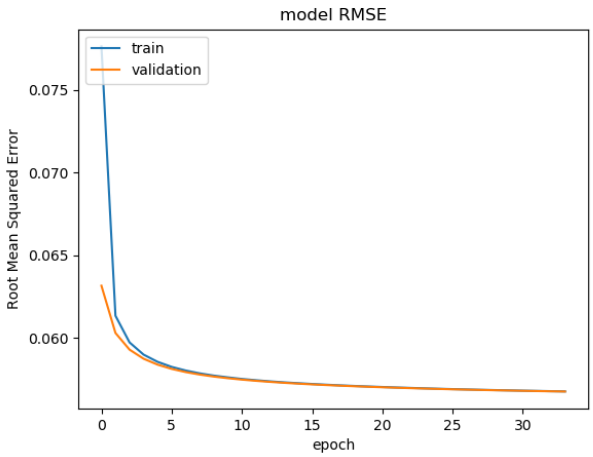

Figura A.34. RMSE de la prova 2.2.9

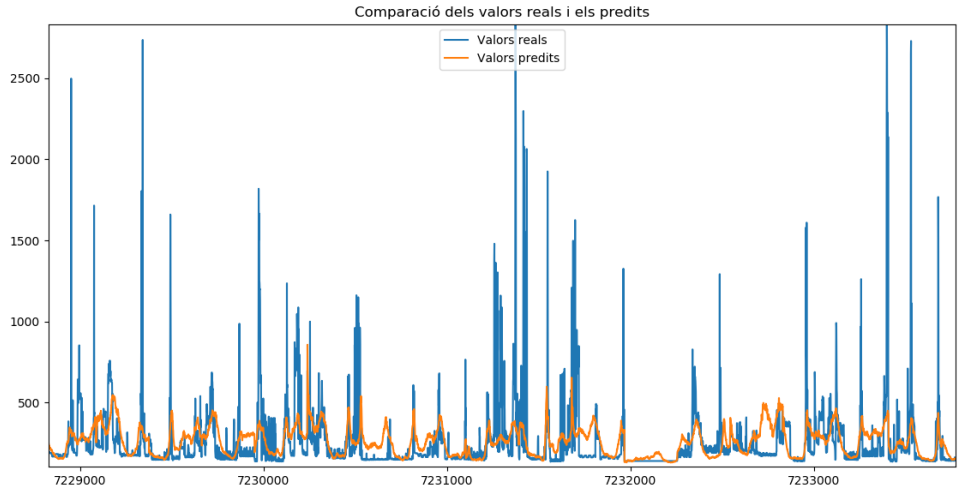

Figura A.35. Prediccions de la prova 2.2.9

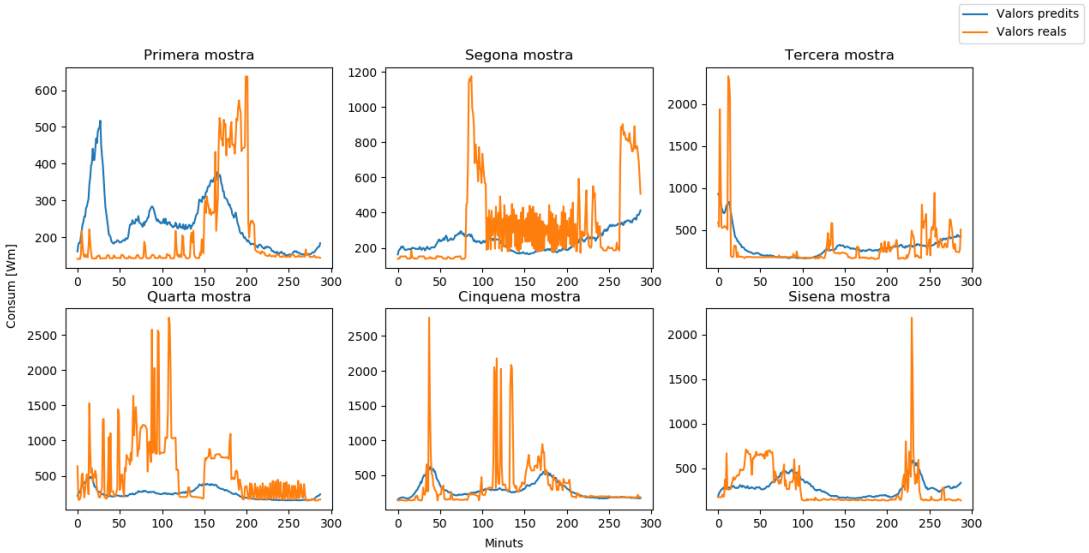

Figura A.36. 6 mostres de la prova 2.2.9

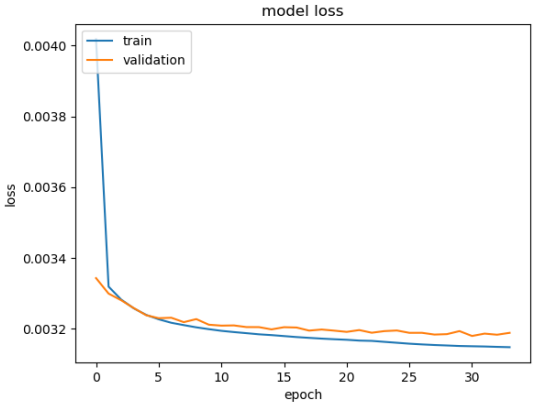

Figura A.37. Loss de la prova 2.2.10

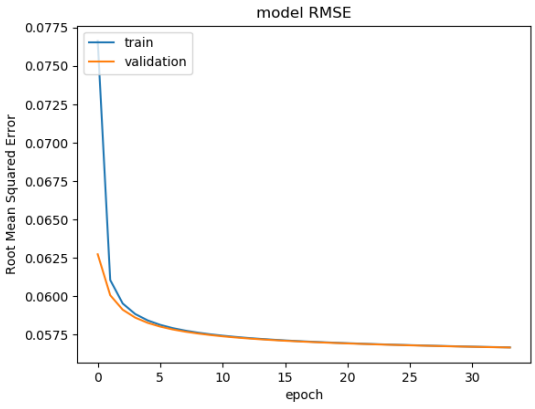

Figura A.38. RMSE de la prova 2.2.10

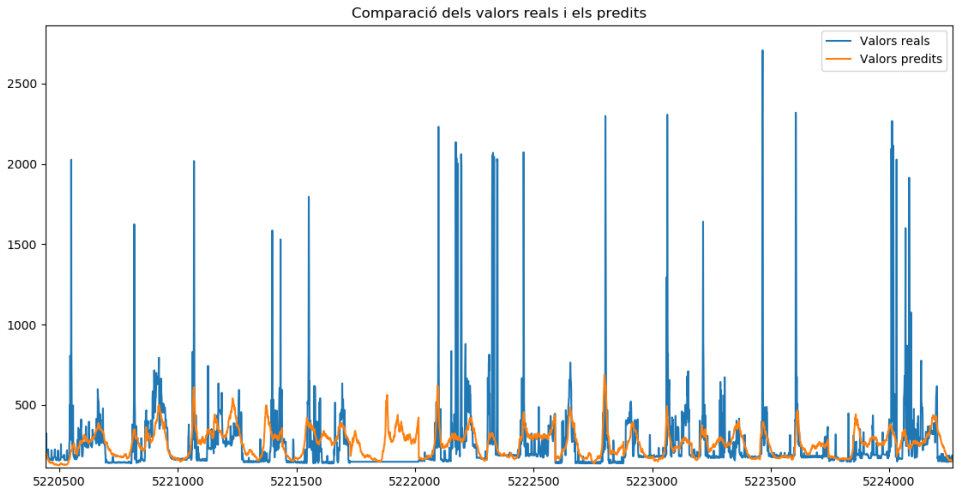

Figura A.39. Prediccions de la prova 2.2.10

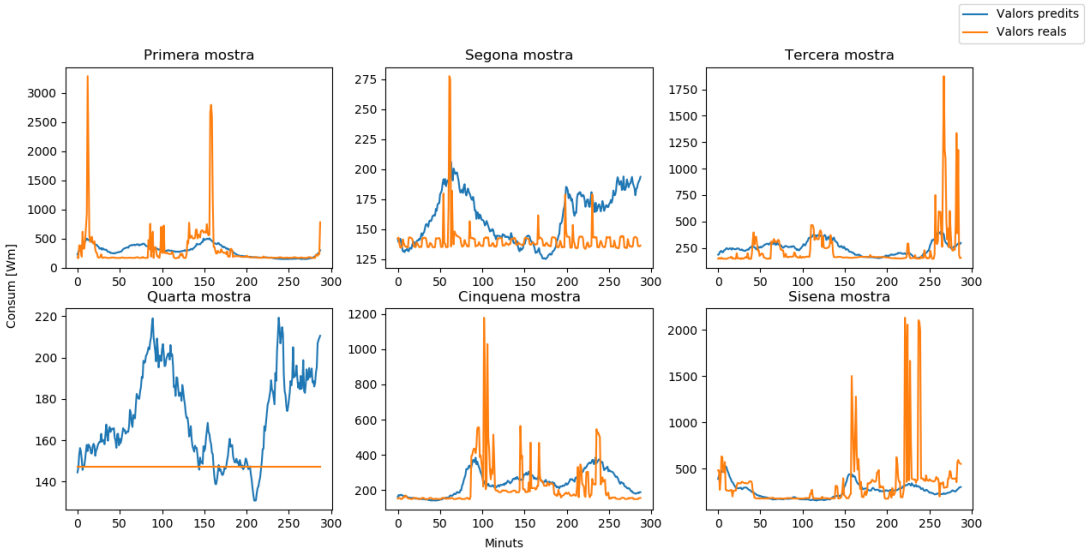

Figura A.40. 6 mostres de la prova 2.2.10

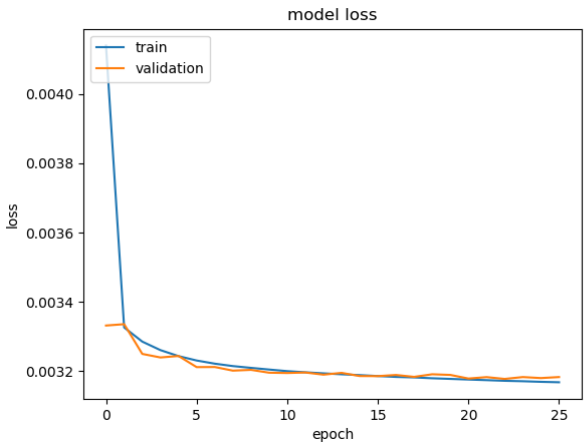

Figura A.41. Loss de la prova 2.2.11

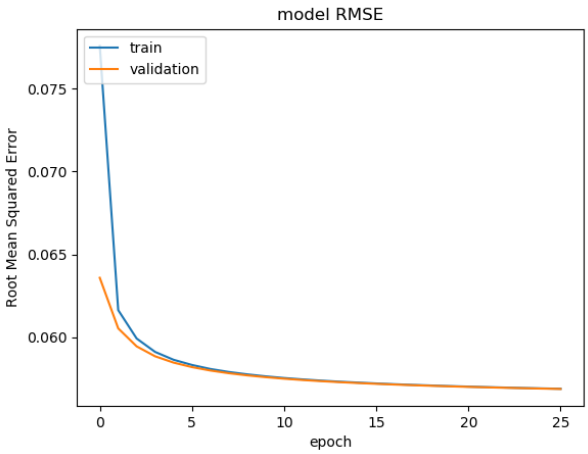

Figura A.42. RMSE de la prova 2.2.11

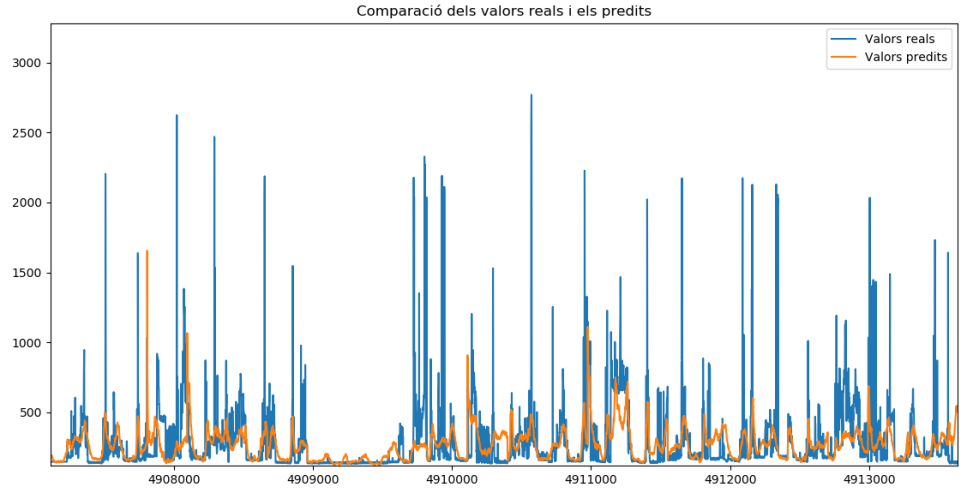

Figura A.43. Prediccions de la prova 2.2.11

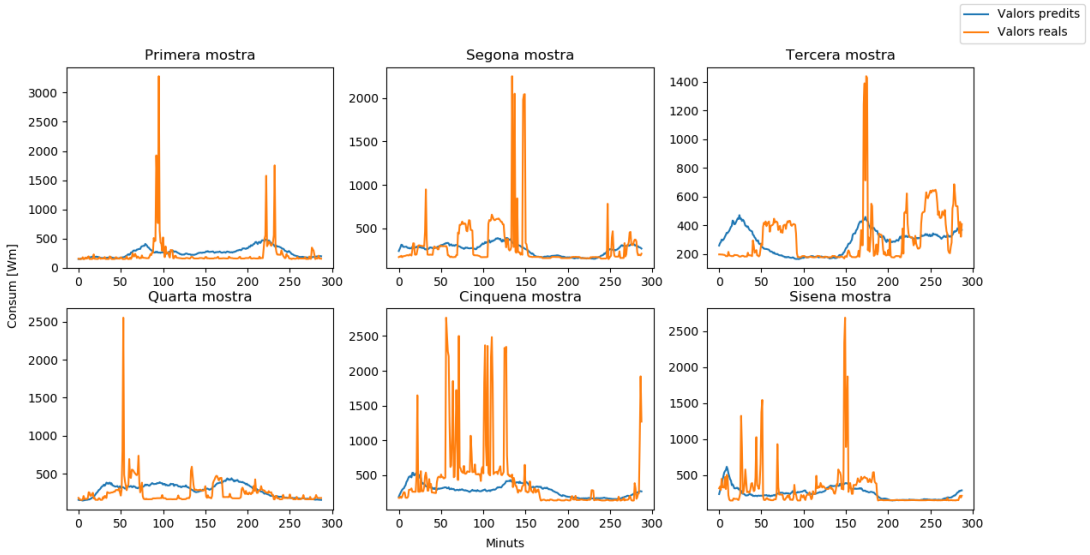

Figura A.44. 6 mostres de la prova 2.2.11

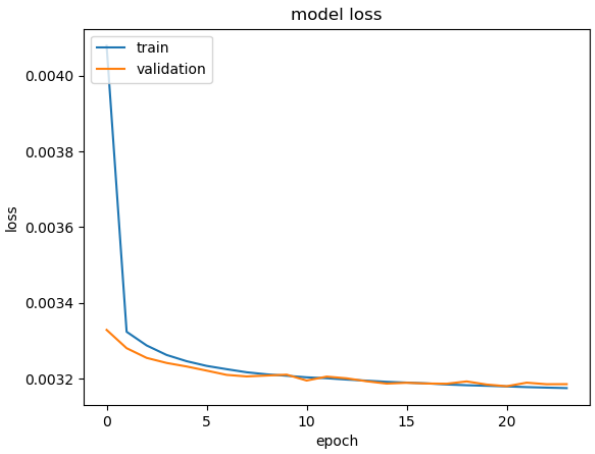

Figura A.45. Loss de la prova 2.2.12

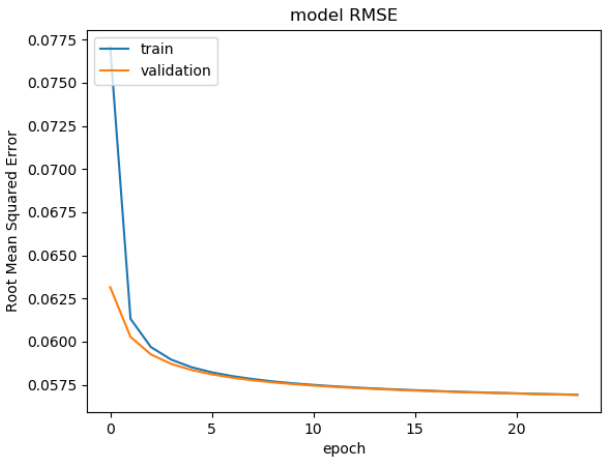

Figura A.46. RMSE de la prova 2.2.12

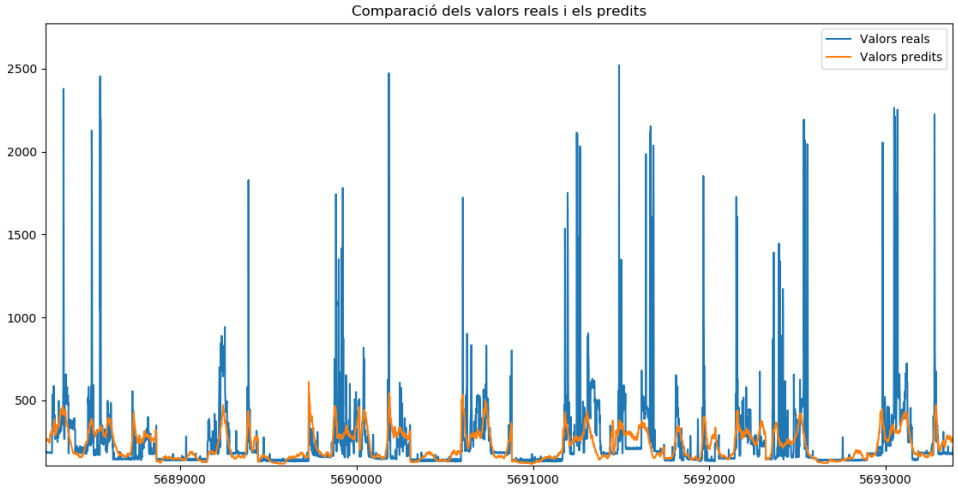

Figura A.47. Prediccions de la prova 2.2.12

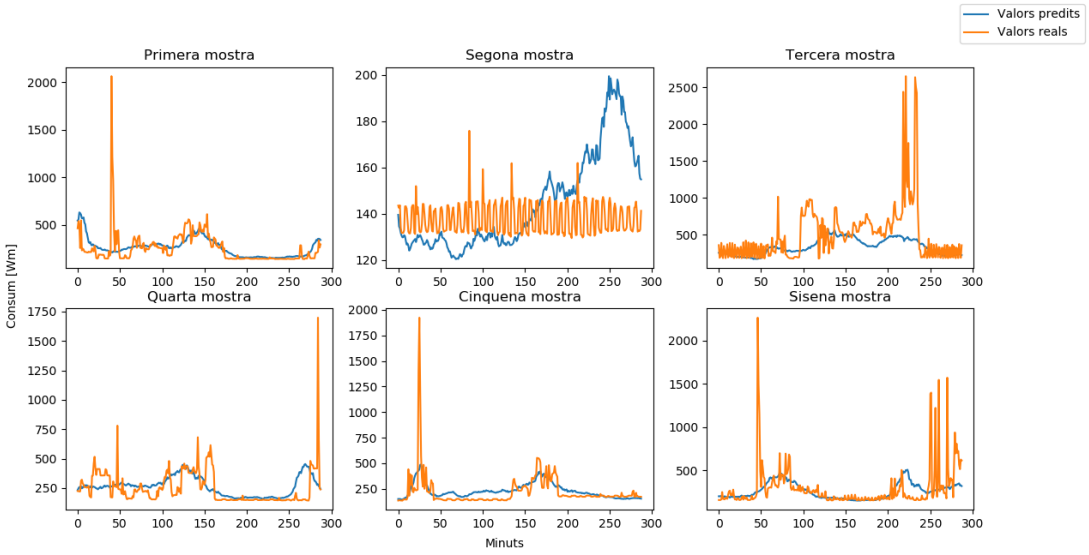

Figura A.48. 6 mostres de la prova 2.2.12

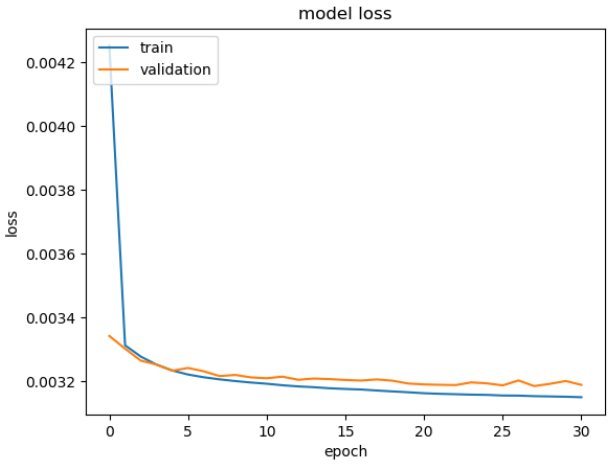

Figura A.49. Loss de la prova 2.2.13

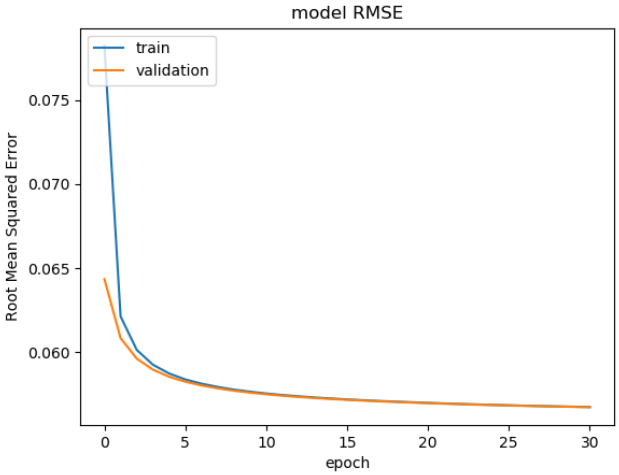

Figura A.50. RMSE de la prova 2.2.13

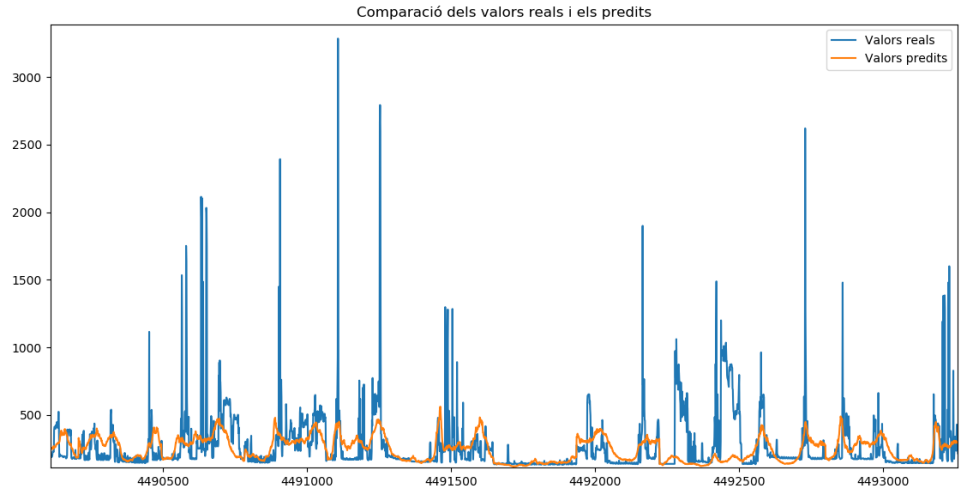

Figura A.51. Prediccions de la prova 2.2.13

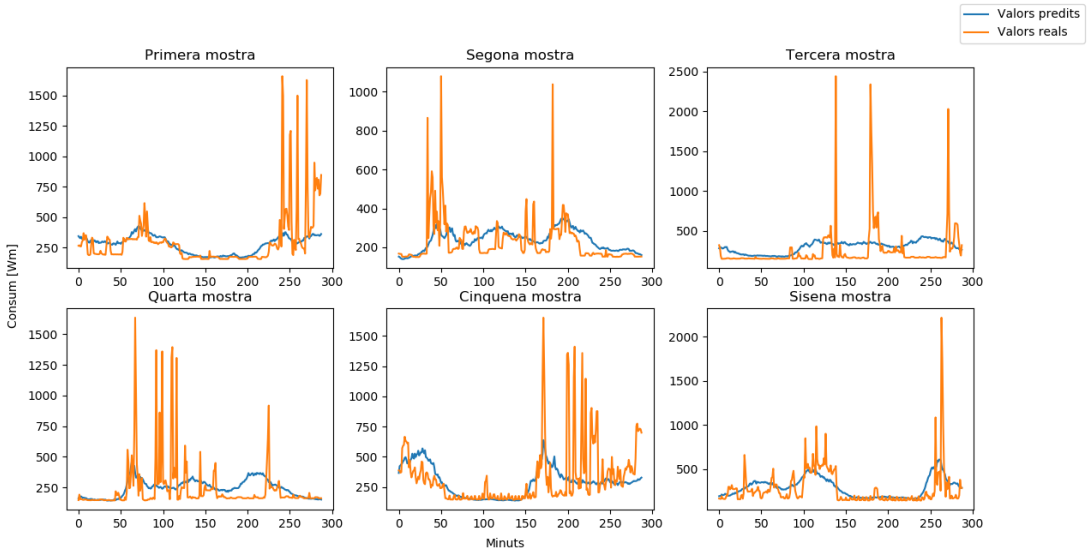

Figura A.52. 6 mostres de la prova 2.2.13

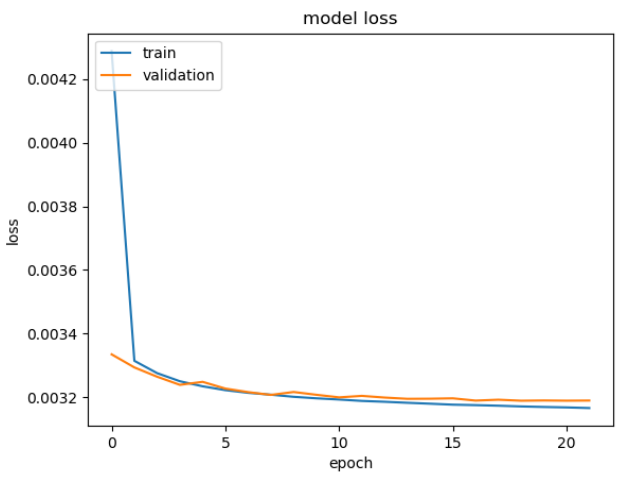

Figura A.53. Loss de la prova 2.2.14

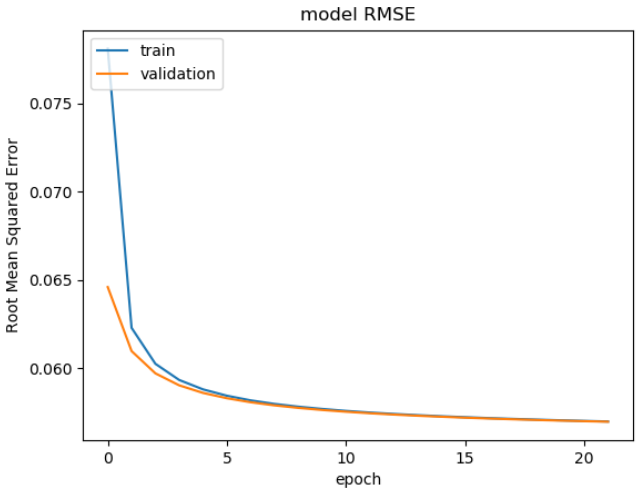

Figura A.54. RMSE de la prova 2.2.14

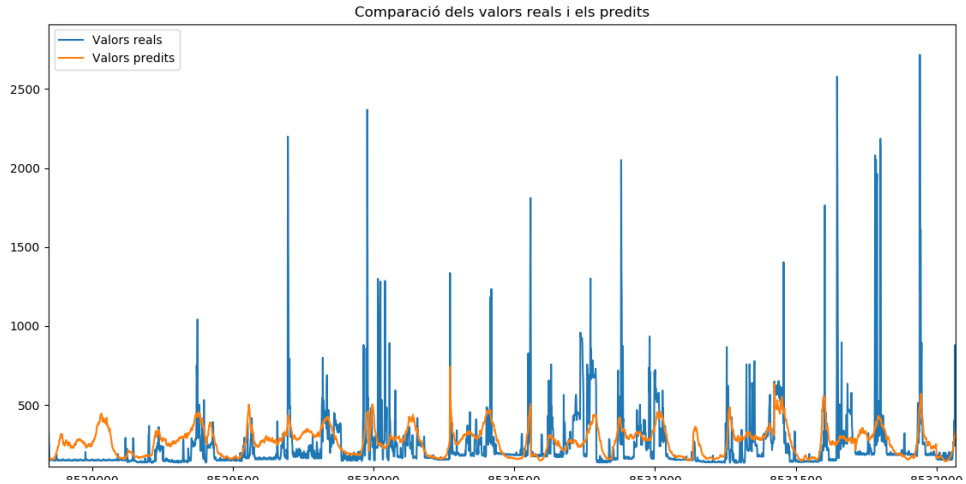

Figura A.55. Prediccions de la prova 2.2.14

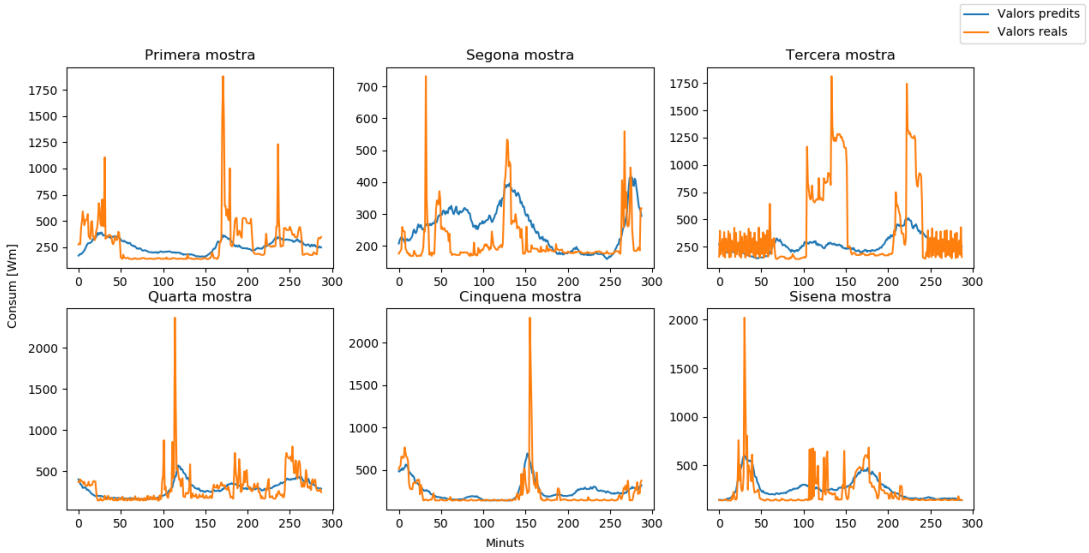

Figura A.56. 6 mostres de la prova 2.2.14

En aquests gràfics s'observa en més detall l'esmentat a la prova 2.2. Quan el nombre de neurones és molt baix, fins a les 150 neurones es pot observar que les prediccions no s'aproximen als valors reals d'una forma tan fidel com amb més neurones. Això està causat perquè, al tenir menys neurones, la xarxa no és capaç d'ajustar els pesos correctament, donant lloc a un problema d'*underfitting.* A partir d'aquest nombre, els resultats van millorant progressivament, i a partir de les 550 neurones, tot i que els valors de la mètrica empitjoren, a les prediccions no és tan visible aquesta diferència. En alguns dels dies que apareixen als gràfics amb 6 mostres s'observa que les prediccions semblen ser molt dolentes: en realitat aquesta diferència no és tan excessiva, sinó que es veu amplificada per l'escala de l'eix de consum. A més, es tracta de dies amb un perfil de consum clarament atípic, per tant, no representen un problema ni una preocupació sobre la qualitat dels resultats, ja que aquestes situacions particulars són pràcticament impossibles de predir.

- **B. Prova 2.3**

Nombre de neurones a cada capa	Test loss	Test RMSE	Iteracions
50	0.00322	0.05706	73
100	0.00311	0.05603	93
150	0.00306	0.05567	67
200	0.00297	0.05475	120
250	0.00295	0.05492	75
300	0.00291	0.05503	60
350	0.00288	0.05424	113
400	0.00542	0.07385	2

Taula B.1. Dades del conjunt de test de la prova 2.3

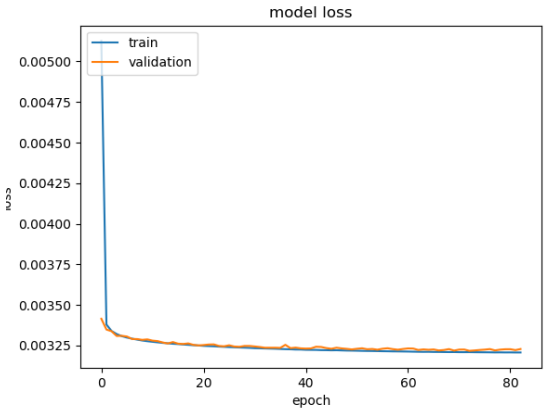

Figura B.1. Loss de la prova 2.3.1

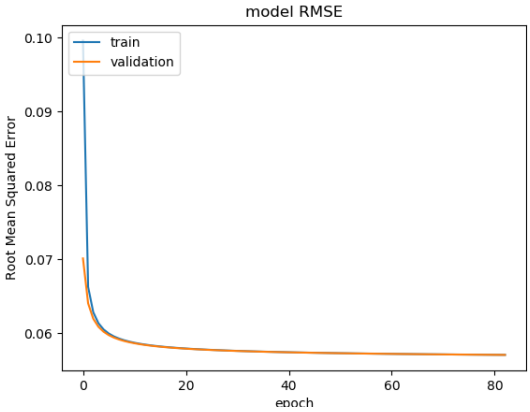

Figura B.2. RMSE de la prova 2.3.1

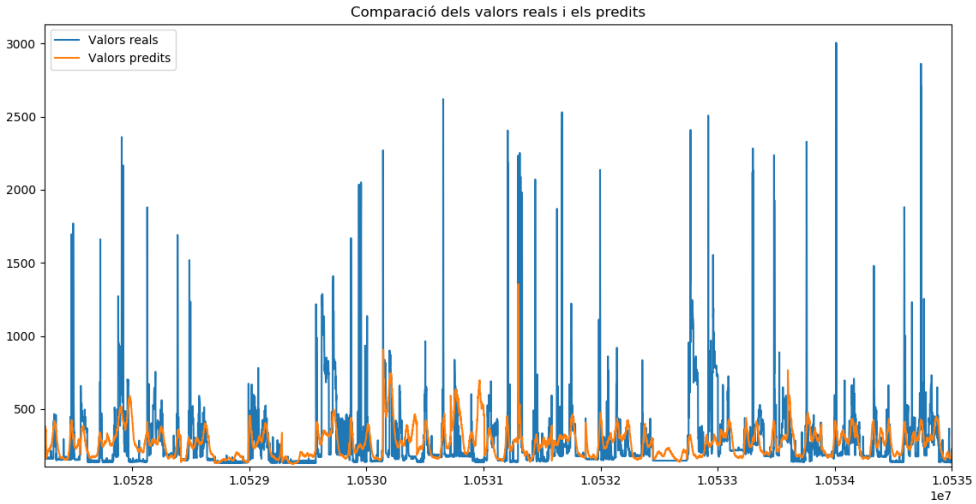

Figura B.3. Prediccions de la prova 2.3.1

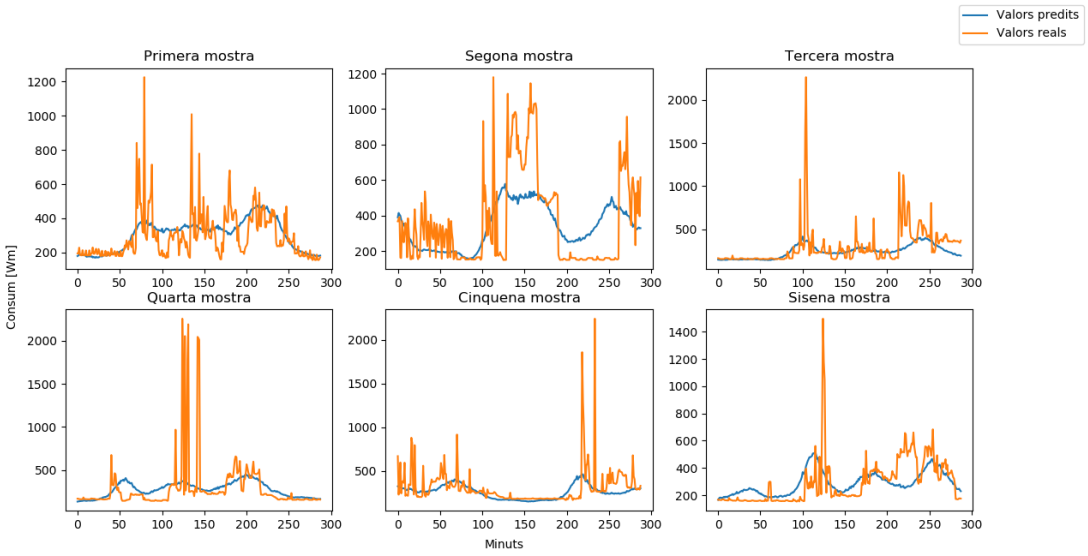

Figura B.4. 6 mostres de la prova 2.3.1

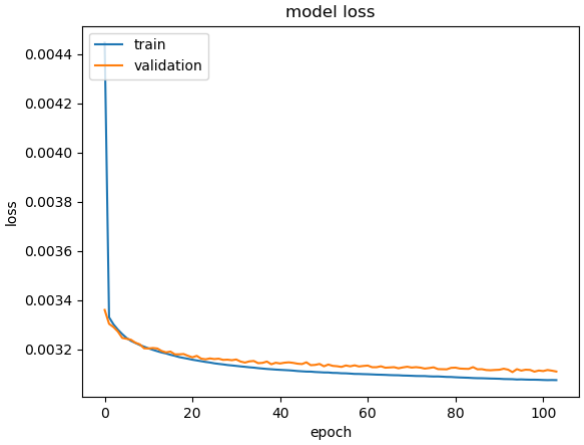

Figura B.5. Loss de la prova 2.3.2

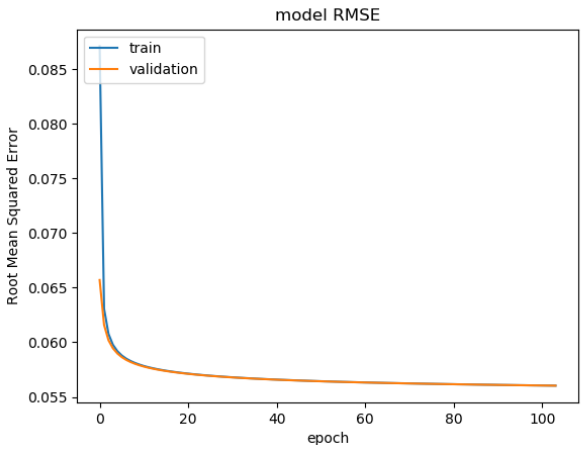

Figura B.6. RMSE de la prova 2.3.2

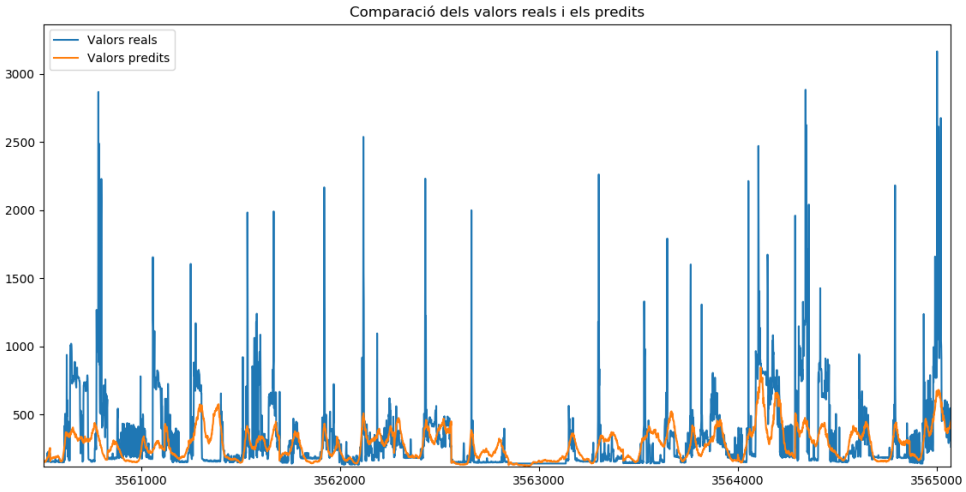

Figura B.7. Prediccions de la prova 2.3.2

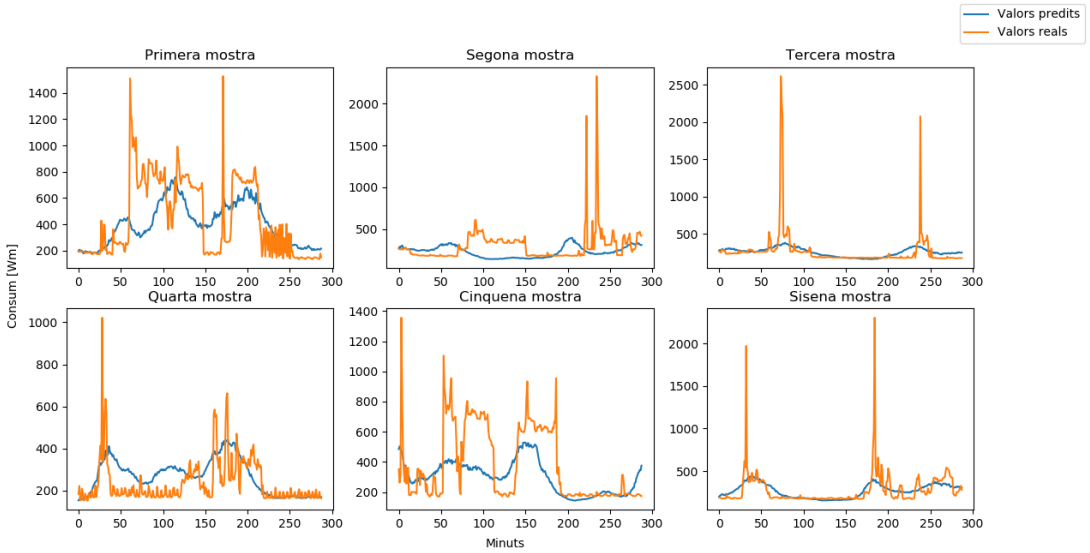

Figura B.8. 6 mostres de la prova 2.3.2

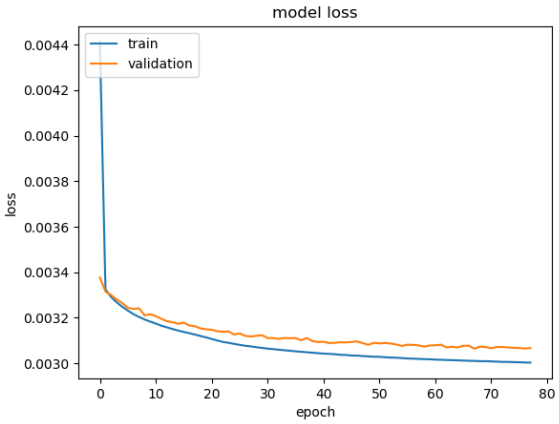

Figura B.9. Loss de la prova 2.3.3

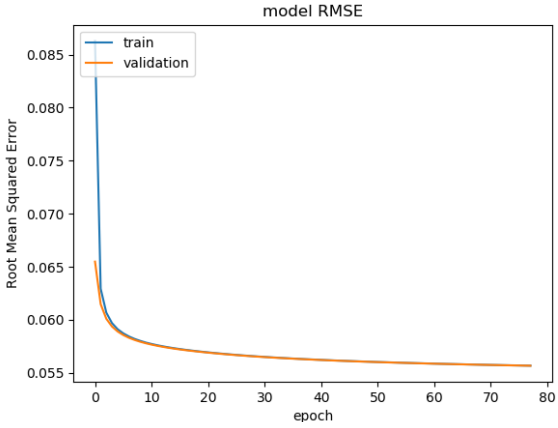

Figura B.10. RMSE de la prova 2.3.3

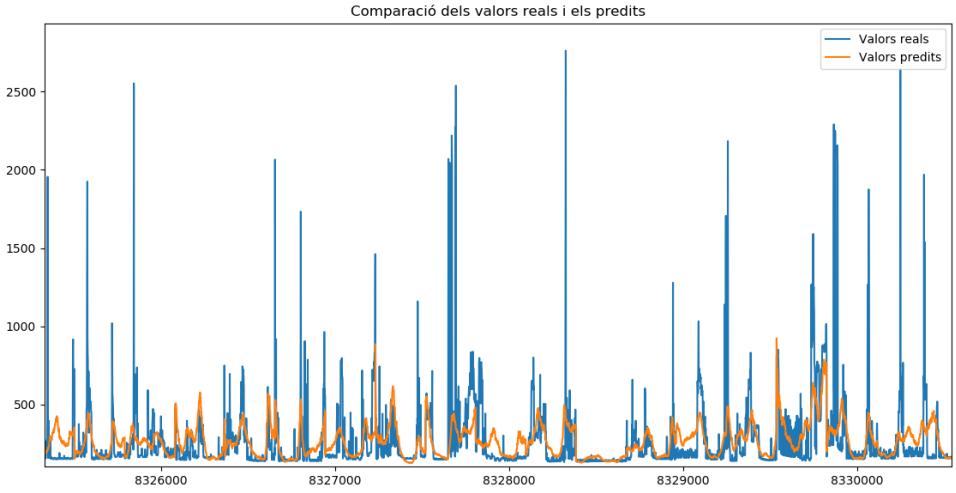

Figura B.11. Prediccions de la prova 2.3.3

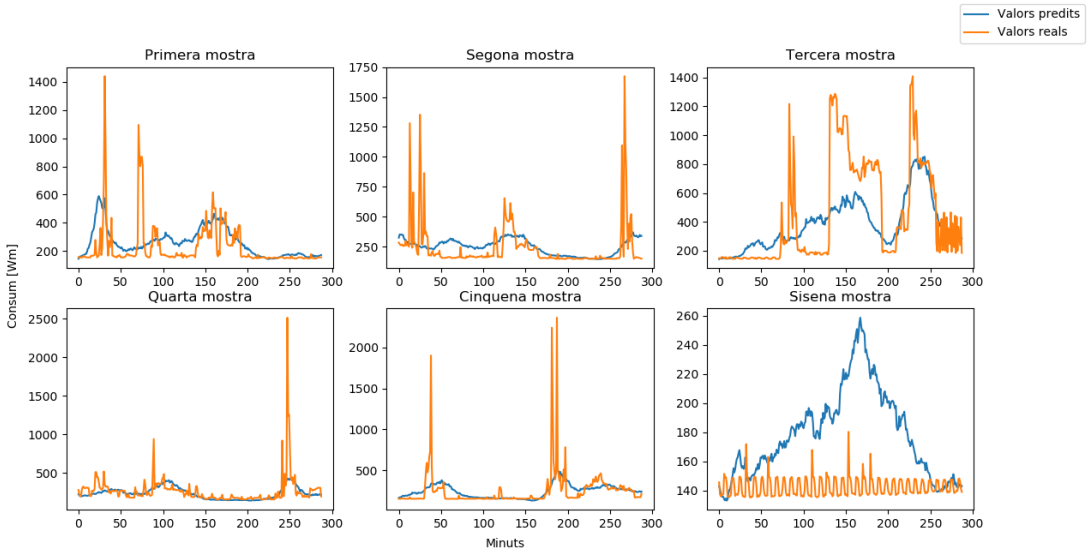

Figura B.12. 6 mostres de la prova 2.3.3

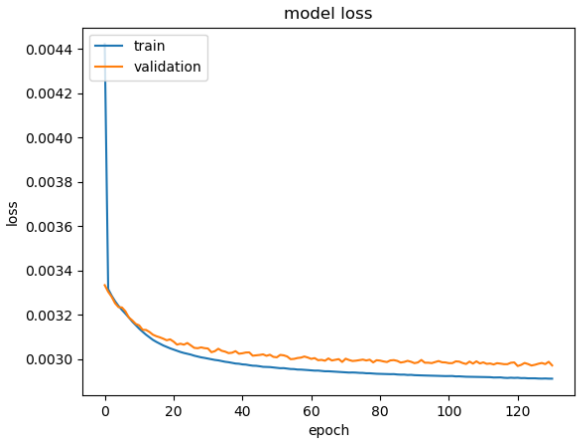

Figura B.13. Loss de la prova 2.3.4

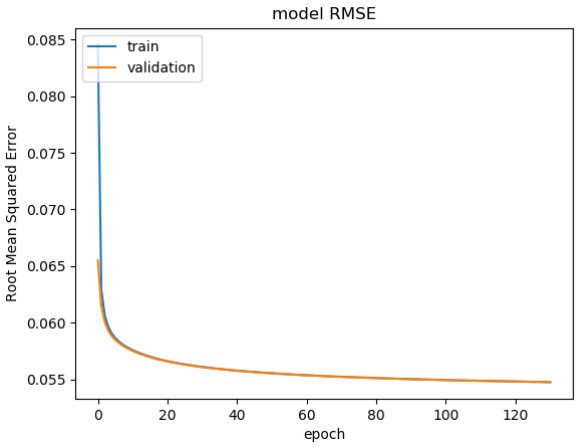

Figura B.14. RMSE de la prova 2.3.4

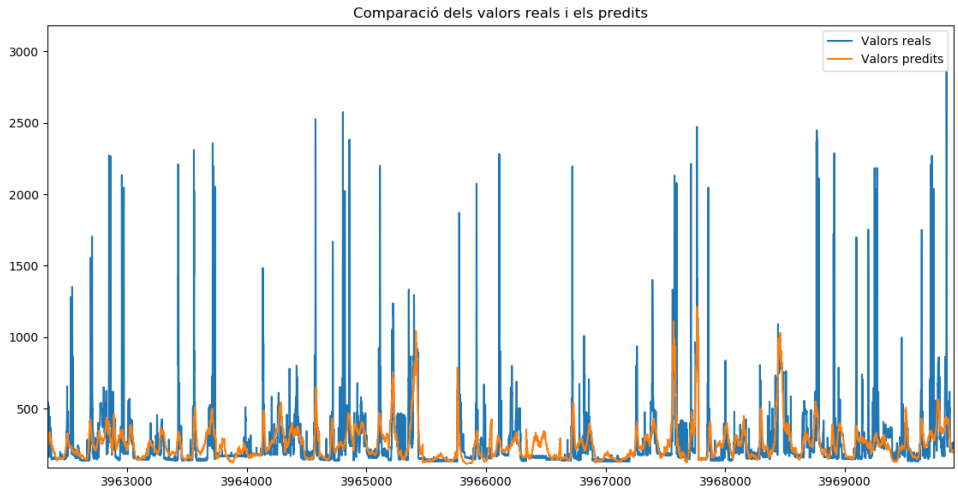

Figura B.15. Prediccions de la prova 2.3.4

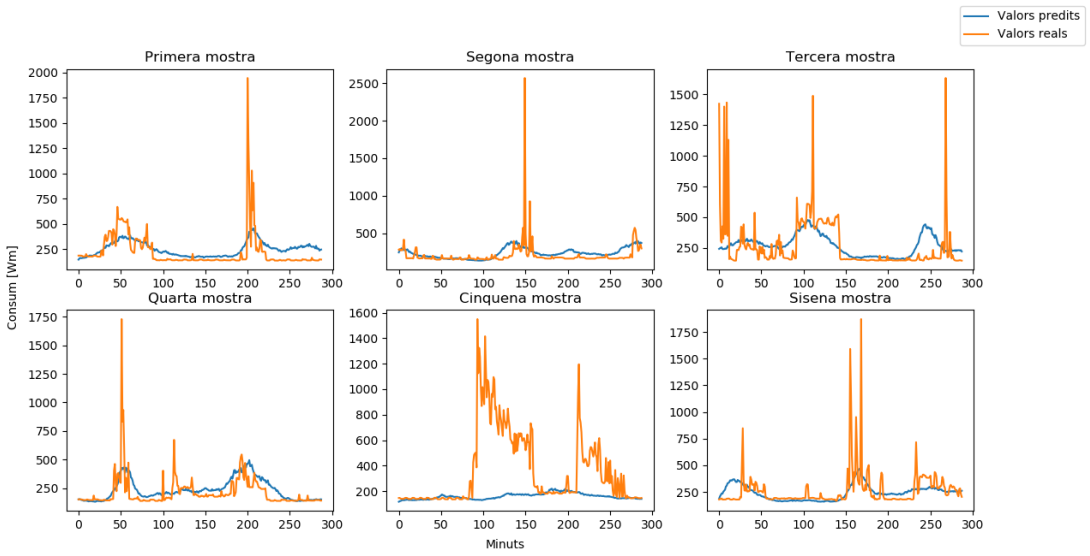

Figura B.16. 6 mostres de la prova 2.3.4

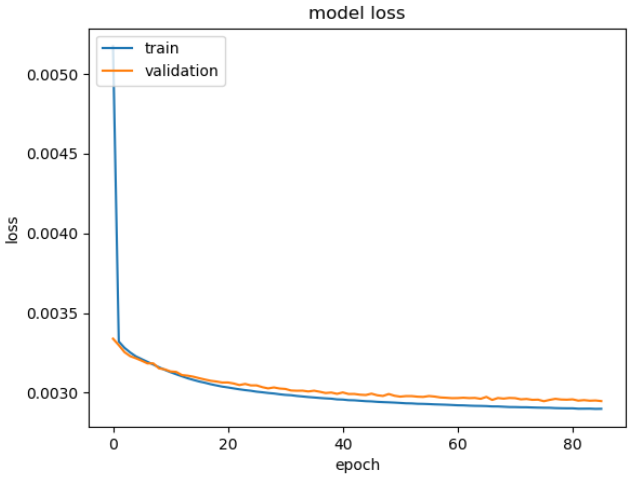

Figura B.17. Loss de la prova 2.3.5

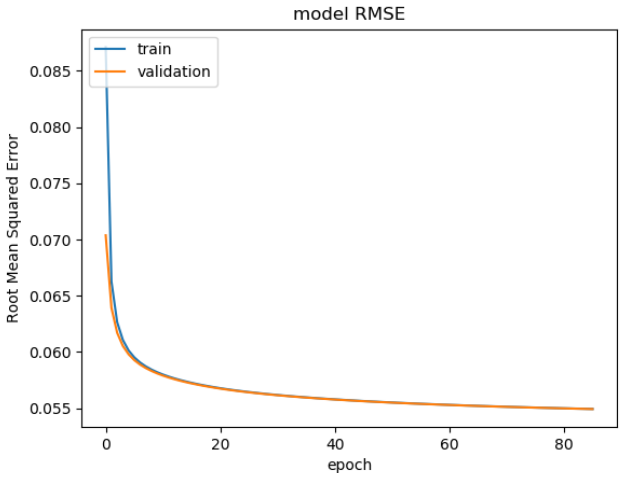

Figura B.18. RMSE de la prova 2.3.5

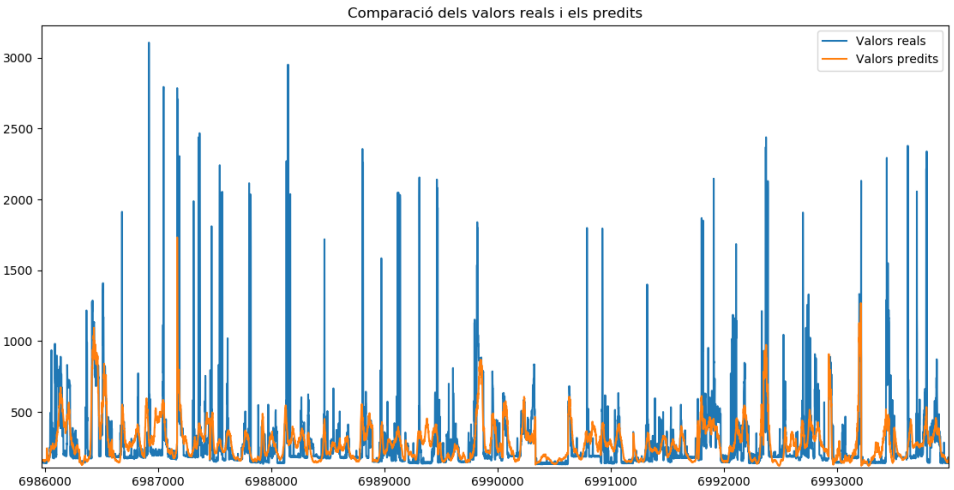

Figura B.19. Prediccions de la prova 2.3.5

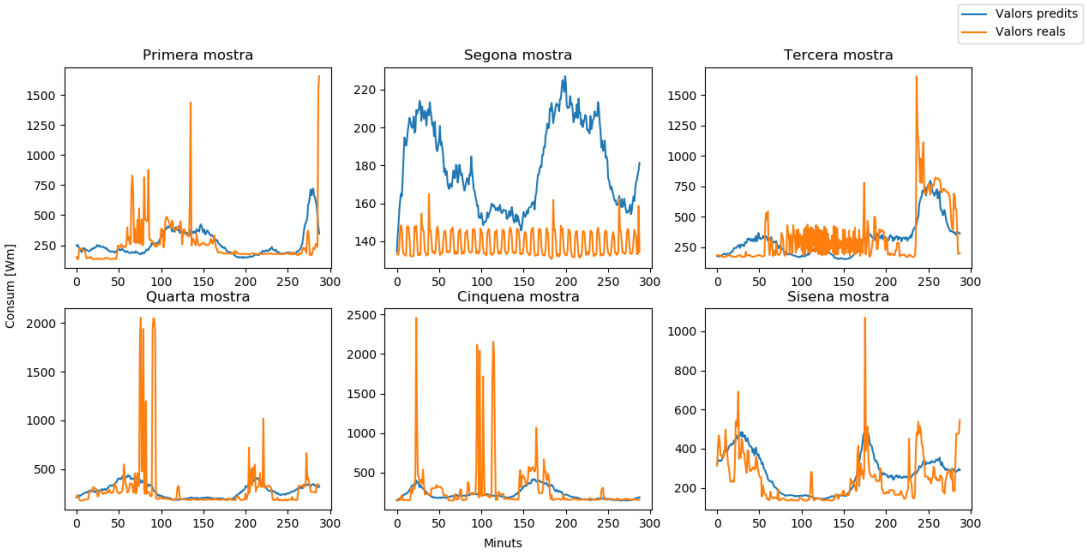

Figura B.20. 6 mostres de la prova 2.3.5

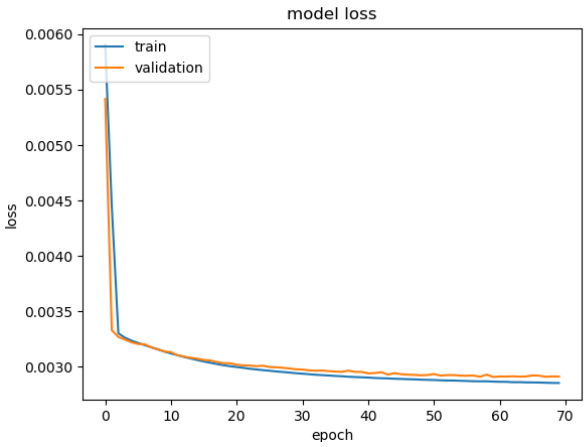

Figura B.21. Loss de la prova 2.3.6

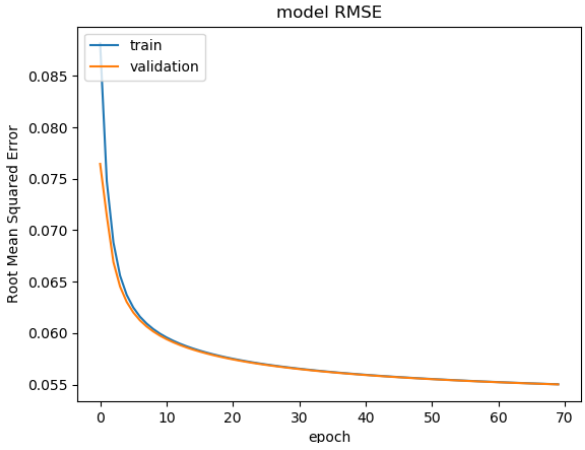

Figura B.22. RMSE de la prova 2.3.6

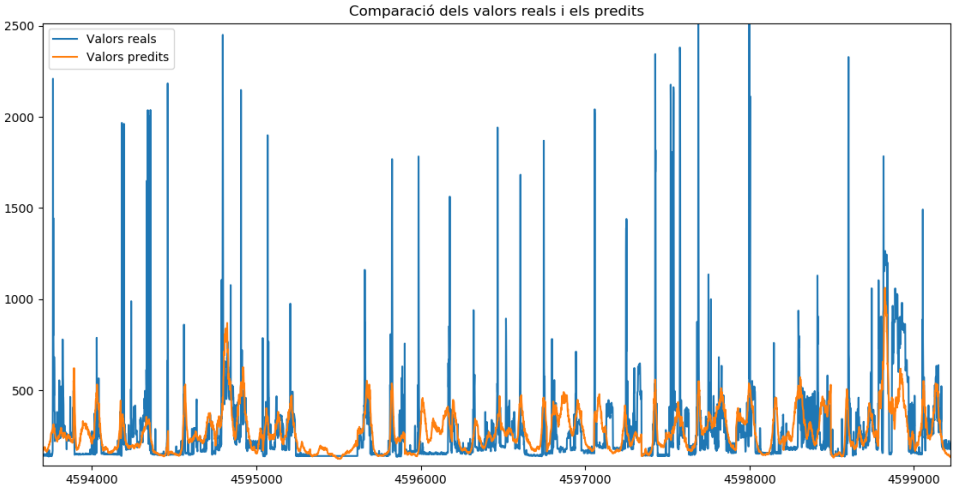

Figura B.23. Prediccions de la prova 2.3.6

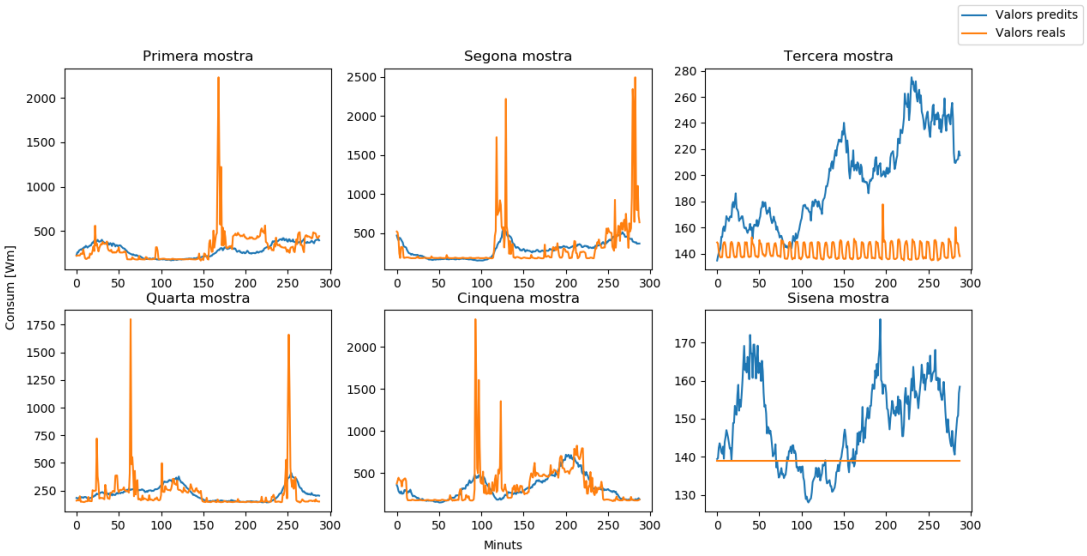

Figura B.24. 6 mostres de la prova 2.3.6

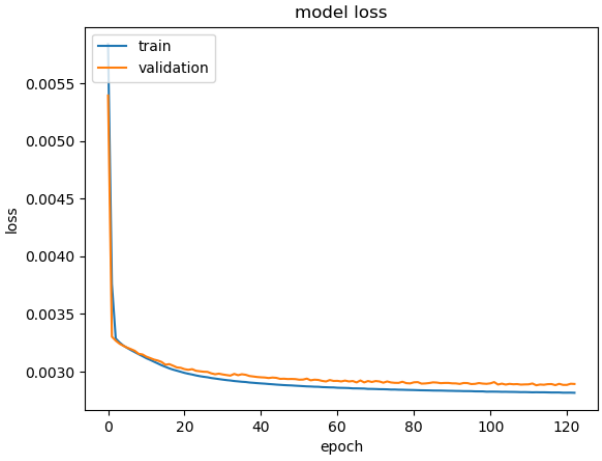

Figura B.25. Loss de la prova 2.3.7

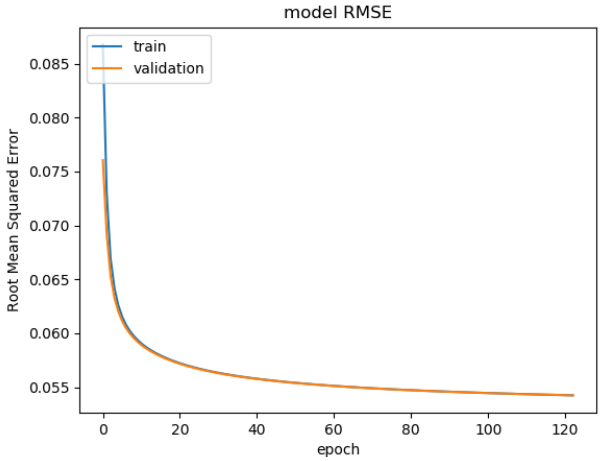

Figura B.26. RMSE de la prova 2.3.7

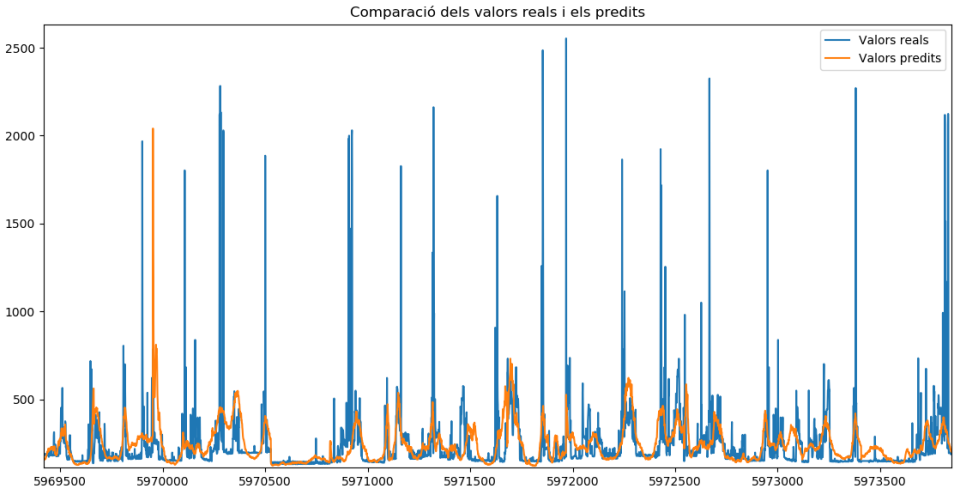

Figura B.27. Prediccions de la prova 2.3.7

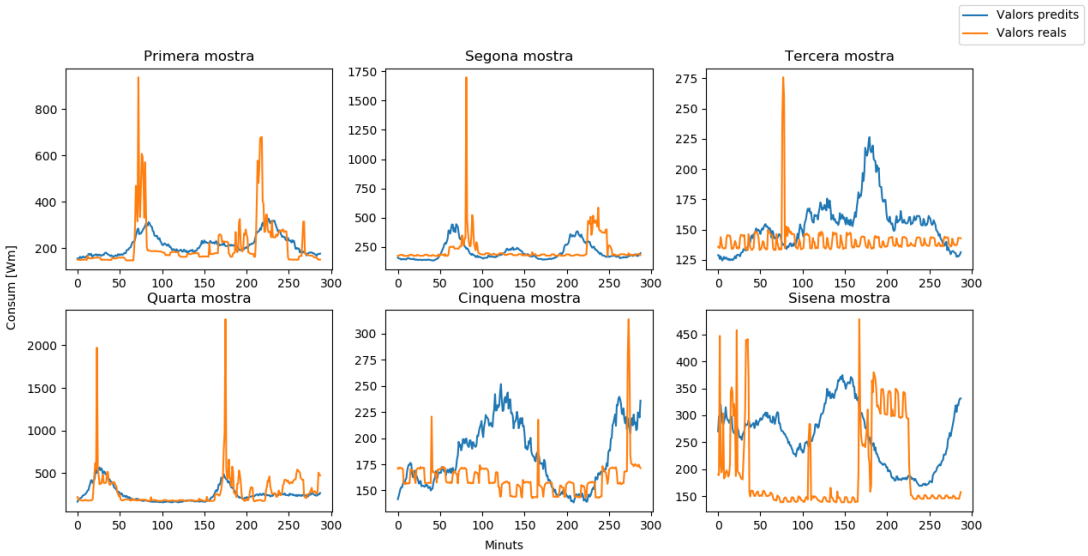

Figura B.28. 6 mostres de la prova 2.3.7

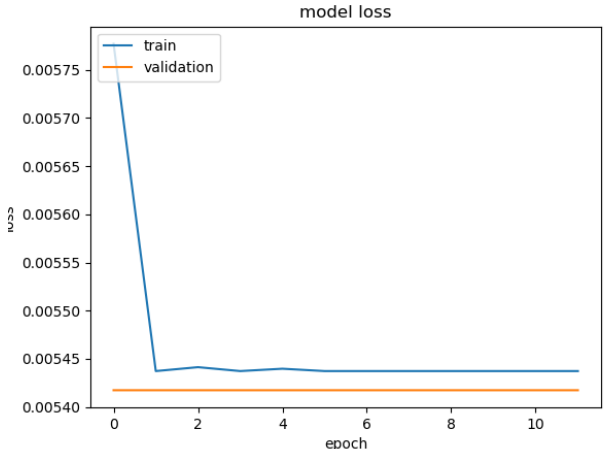

Figura B.29. Loss de la prova 2.3.8

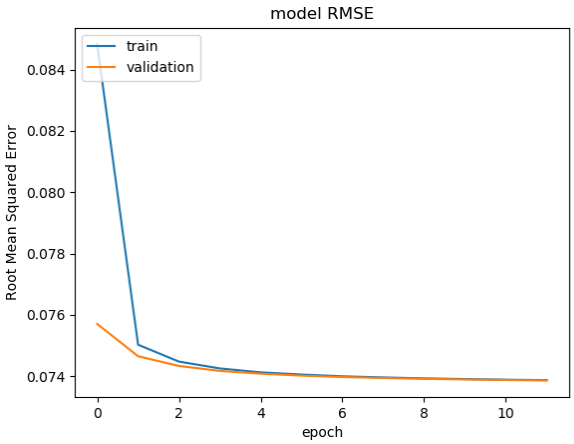

Figura B.30. RMSE de la prova 2.3.8

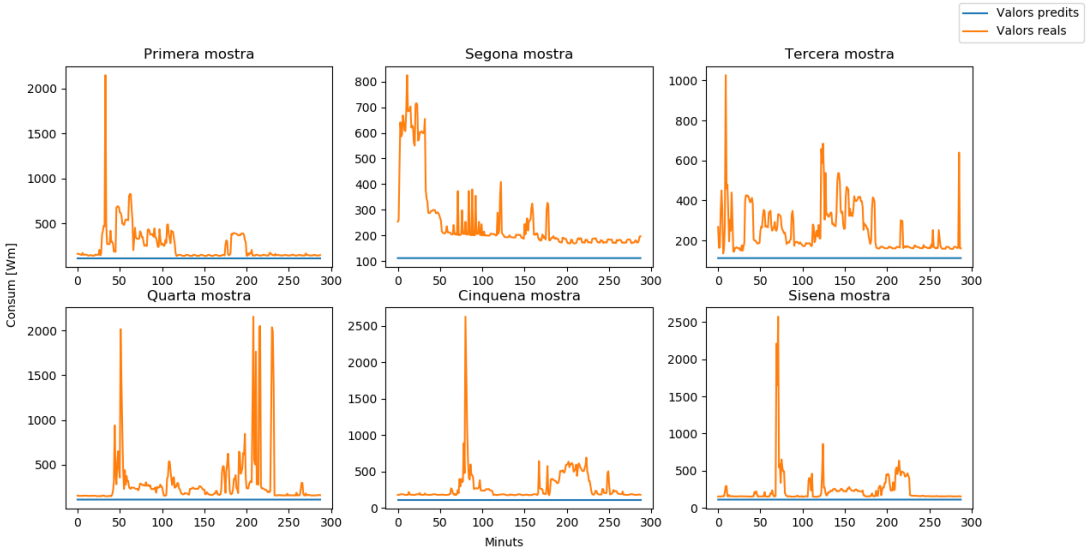

Figura B.31. 6 mostres de la prova 2.3.8

Es pot observar que les prediccions milloren quan s'incrementa el nombre de neurones. El nombre d'iteracions necessàries a cada una de les etapes s'incrementa significativament respecte a quan es tenia una sola capa oculta, fet també causat per l'increment de l'argument *patience,* que marca el final de l'entrenament, de 3 fins a 10 iteracions sense canvis o amb un increment en l'error del conjunt de validació. Tot i això, hi ha una clara millora respecte a la prova 2.2, per tant, com s'ha dit abans, aquesta és l'estructura a triar a l'hora de realitzar les prediccions.

A les figures B.29 i B.30 s'observa amb més claredat el risc que comporta una estructura de les capes ocultes excessivament complexa, que causa el fenomen de *dying ReLU* esmentat al nucli de la memòria: la xarxa arriba a un punt al qual esdevé incapaç d'activar totes les seves neurones a causa de la pròpia complexitat d'aquesta, per tant, totes les funcions d'activació mantenen les neurones de les capes ocultes en un estat inactiu.